Numéro d'ordre : DI 39

THESES

présentées à

la Faculté des Sciences

de l'Université de Clermont-Ferrand

pour obtenir

le Titre de Docteur-Ingénieur

en Informatique

par

René BOUDEAUD

Ingénieur C.N.A.M.

Première Thèse :

ETUDE ET REALISATION D'UN COMPILATEUR ALGOL POUR

CALCULATEUR PB 250 - A 100

Deuxième Thèse

Proposition donnée par la Faculté :

MISE AU POINT DE LA CONNEXION D'UNE PLATINE DE

MICROSCOPE A UN PERFORATEUR DE RUBANS

Soutenues le 6 Décembre 1967 devant la Commission d'Examen

MM. AVAN	Président
HENNEQUIN	Examinateurs
BOLLIET	

Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41546606_0013

Numéro d'ordre : DI 39

T H E S E S

présentées à

la Faculté des Sciences

de l'Université de Clermont-Ferrand

pour obtenir

le Titre de Docteur-Ingénieur

en Informatique

par

René BOUDEAUD

Ingénieur C.N.A.M.

Première Thèse :

ETUDE ET REALISATION D'UN COMPILATEUR ALGOL POUR

CALCULATEUR PB 250 - A 100

Deuxième Thèse

Proposition donnée par la Faculté :

MISE AU POINT DE LA CONNEXION D'UNE PLATINE DE

MICROSCOPE A UN PERFORATEUR DE RUBANS

Soutenues le 6 Décembre 1967 devant la Commission d'Examen

MM. AVAN	Président
HENNEQUIN	Examinateurs
BOLLIET	

UNIV.
LEIDEN
BIBL.

FACULTE DES SCIENCES

Liste des professeurs

Doyen : M. SAUREL Jean

1er Assesseur : M. KOLHMULLER Robert

PROFESSEURS

M. RAMBAUD René	Chimie
M. BLANC Eugène	Mécanique Rationnelle
M. ROQUES Maurice	Géologie
M. LAPADU-HARGUES Pierre	Minéralogie
M. TRUFFAULT Robert	Chimie Organique
M. LUTZ Hubert	Biologie Animale
M. RAOULT Gaston	Physique
M. CHAMPAGNAT Paul	Physiologie Végétale
M. DUPLESSIS-KERGOMARD Alain	Chimie
M. AVAN Louis	Physiqee Expérimentale
M. PARIAUD Jean-Charles	Chimie Physique
Mme HENNEQUIN Françoise	Mathématiques
Mle BERTHET Ginette	Physique
M. LOISEAU Jean	Botanique
M.COMBE Rané	Physique
M. de PUYTORAC Pierre	Zoologie
Mme AVAN Madeleine	Physique
Mle DONDON Marie-Louise	Chimie Physique
M. DELOST Paul	Physiologie Animale
M. SAUREL Jean	Physique
M. VIALLARD André	Chimie
M. KOHLMULLER Robert	Chimie Minérale
M. GUILLAUME Marcel	Algèbre
M. COMBE Jean	Physique Générale
M. HENNEQUIN Paul-Louis	Mathématiques Appliquées
M. L'HERITIER	Génétique

PROFESSEURS SANS CHAIRE

M. PETERLONGO Jean-Marc	Géologie
M. BAILLAUD Lucien	Botanique
M. VESSIERE Roger	Chimie
M. BATHIER Marcel	Physique
Mlle DUCHER Suzanne	Chimie
M. FALGAS Maurice	Mathématiques
M. ROCHE Jean-Etienne	Physique
M. MARTINET Pierre	Chimie
M. MAISONNEUVE Jacques	Géologie
M. CAPESTAN Michel	Chimie
M. LENOIR Marcel	Mathématiques
M. THOMAS Jean	Physique

MAITRES DE CONFERENCES

M. BERNARD Maurice	Physique
M. JOYON Louis	Zoologie
M. ISABELLE Didier	Physique
M. CAUBERE Paul	Chimie
M. SOULAGE René	Géophysique Générale
M. DUFAURE Jean-Paul	Biologie Animale
M. COURDUROUX Jean-Claude	Zoologie
M. ARIBAUD François	Mathématiques

CHARGES D'ENSEIGNEMENT

M. CHADEYRAS Marcel	Mathématiques
M. BADRIKIAN Albert	Mathématiques
Mme BESSERRE Anne-Marie	Mathématiques
M. BOILLAT Guy	Mathématiques
M. LE THANH Phong	Mathématiques
M. JEAN Claude	Physiologie Animale

ATTACHEE PRINCIPALE

Mme CHOPART Marie-Louise

Je tiens à exprimer ma profonde reconnaissance

A Monsieur P.L. HENNEQUIN, Directeur du Laboratoire de Calcul de la Faculté des Sciences de Clermont-Ferrand, qui m'a permis de réaliser ce travail et dont la confiance et les conseils m'ont été indispensables pour le mener à bien.

A Monsieur L. AVAN, Directeur du Laboratoire de Physique Nucléaire de la Faculté des Sciences de Clermont-Ferrand, qui a bien voulu me faire l'honneur de présider le Jury et a proposé le sujet de ma deuxième thèse.

A Monsieur L. BOLLIET, Maître de Conférences a la Faculté des Sciences de Grenoble, qui a accepté de faire partie du Jury et s'est intéressé à mon travail.

A Monsieur J. KUNTZMANN, Directeur de l'Institut de Mathématiques Appliquées de Grenoble, pour la bienveillante sollicitude qu'il m'a témoignée et les précieux conseils qu'il m'a prodigués.

Je tiens également à remercier tout le personnel du Laboratoire de Calcul de la Faculté des Sciences de Clermont-Ferrand, en particulier :

Messieurs LAMBEY,
BERTHON
Mesdames BARTHOMEUF,
COURAGEOT

qui m'ont aidé dans la réalisation matérielle de ce travail.

A mon père

A la mémoire de ma mère

TABLE DES MATIERES

	Pages
INTRODUCTION	1
PREMIERE PARTIE : THEORIE ALGEBRIQUE DES LANGAGES	
- Monoïde libre	3
- Ensemble des parties d'un monoïde libre	6
- Varaibles, expressions, fonctions, équations dans l'algèbre du gerbier libre	12
- Langages de programmation	16
DEUXIEME PARTIE : RECONNAISSANCE D'UN LANGAGE	
- Automate de reconnaissance d'un langage	21
- Etude des équations syntaxiques d'ALGOL 60	41
- Schémas de l'automate de reconnaissance associé au système d'équations syntaxiques simplifié d'ALGOL 60	67
TROISIEME PARTIE : TRADUCTION D'UN LANGAGE	
- Le Problème de la traduction	91
- Traduction des expressions	93
- Traduction des instructions	112
- Déclarations	124
CONCLUSION	131
ANNEXE :	
- Exemples de programmes d'analyse numérique utilisables sur PB 250	

INTRODUCTION

Le but de notre travail a été l'étude et la réalisation d'un "compilateur" d'ALGOL 60 sur calculateur PB 250, type A 100, SETI.

On entend par "compilateur", un programme qui permet d'interpréter un texte écrit dans un langage symbolique tel qu'ALGOL en un programme écrit en langage machine.

On appelle aussi un tel programme, un "traducteur".

Les difficultés que nous avons rencontrées étaient dues au calculateur PB 250 ; celui-ci ne possède qu'une capacité mémoire relativement faible et ses instructions sont à simple adresse et sans adressage indirect.

La méthode que nous avons utilisée doit donner des résultats intéressants sur les calculateurs récents. Néanmoins, le PB 250 possède une programmation symbolique MAGE comparable à FORTRAN. Des essais effectués à la traduction d'un texte ALGOL et de son équivalent MAGE, il ressort un rapport des temps de traduction de 1,5. (Précisons que nous n'avons pas optimisé entièrement le traducteur, faute de temps, de place et à cause de la constitution même de la mémoire du PB 250).

Pour obtenir la possibilité de construire ce traducteur sur PB 250, il nous a fallu réduire les règles syntaxiques.

Celles-ci ont été interprétées comme un système d'équations dans l'algèbre des parties d'un monoïde libre. Elles se sont simplifiées de façon intéressante. De 98 équations extraites du rapport ALGOL 60, on en déduit 29 directement utilisables dans le traducteur.

Celui-ci est constitué d'un assemblage de sous-programmes de deux types :

1 - Sous-programmes de lecture de texte ALGOL

2 - Sous-programmes de génération de langage machine

Ces derniers sont connectés conformément aux 29 règles syntaxiques.

PREMIERE PARTIE

THEORIE ALGEBRIQUE DES LANGAGES

MONOIDE LIBRE

Définitions : [cf : 1, 2, 3 et 4]

Soit un ensemble auxiliaire non vide X dont les éléments x_i, $i = 1, 2, \ldots n$ sont appelés "éléments libres".

Cet ensemble est appelé "système de générateurs libres" du monoïde.

Un "mot" est un système ordonné fini d'éléments de l'ensemble X, avec des répétitions quelconques.

Il peut être écrit sous la forme :

$$\boxed{W = x_1 \; x_2 \ldots x_n \qquad n \geqslant 0}$$

On appelle longueur d'un mot le nombre d'éléments qu'il contient.

L'ensemble des mots composés d'un seul élément libre est isomorphe à l'ensemble X .

Un "monoïde libre" est l'ensemble de tous les mots formés à partir d'un système de générateurs libres X .

On peut écrire cet ensemble sous la forme :

$$\boxed{M = \{W_1, W_2, W_3, \ldots\}}$$

Loi de composition interne dans un monoïde libre

On définit et on note par un point "." , l'opération de "concaténation" que l'on peut aussi appeler "produit libre", dans l'ensemble M par :

$\forall W_1 \in M$, $\forall W_2 \in M$ on a $W_1 . W_2 \in M$

$W_1 . W_2$ est obtenu en juxtaposant le mot W_2 à droite du mot W_1.

Si $W_1 = x_1 . x_2 \dots x_p$ $\quad p \geq 0$

$W_2 = y_1 . y_2 \dots y_q$ $\quad q \geq 0$

On a : $W_1 . W_2 = x_1 . x_2 \dots x_p . y_1 . y_2 \dots y_q$ $\quad p, q \geq 0$

Elément neutre

Il existe un élément neutre I pour le produit libre appartenant à M tel que :

pour tout $W \in M$, on ait :

$$\boxed{W . I = I . W = W}$$

Cet élément neutre est appelé le "mot vide" .
C'est le seul mot de longueur nulle.

Propriété d'associativité du produit libre

On définit l'associativité du produit libre par :

$\forall W_1, W_2, W_3 \in M$

On a : $W_1 . (W_2 . W_3) = (W_1 . W_2) . W_3$

Nota

En général le produit n'est pas commutatif

Remarque

Un monoïde libre est donc un ensemble muni d'une opération associative appelée "concaténation", possédant un élément neutre.

Simplification des mots

Soit le mot $W = x_1 \cdot x_1 \cdot x_2 \cdot x_2 \cdot x_2 \cdot x_3$

L'utilisation des puissances d'éléments permet d'écrire plus brièvement ce mot sous la forme :

$$W = x_1^2 \cdot x_2^3 \cdot x^3$$

Exemple de monoïde libre

Soit l'ensemble $X = \{0, 1\}$

M : le monoïde libre relativement au système de générateurs X, est l'ensemble de toutes les représentations binaires des entiers positifs.

ENSEMBLE DES PARTIES D'UN MONOIDE LIBRE

Définition : [cf : 3]

Etant donné un monoïde libre M , ses sous-ensembles peuvent être considérés comme les éléments d'un ensemble qu'on note $\mathcal{P}(M)$ et que l'on appelle : ensemble des parties du monoïde libre.

Lois de composition interne dans $\mathcal{P}(M)$

On définit dans l'ensemble $\mathcal{P}(M)$, les opérations suivantes :

soit : $G = \mathcal{P}(M) = \{A, B, C \ldots\}$

Somme ou Union

C'est l'union de deux sous-ensembles au sens habituel que l'on notera ici, pour des raisons typographiques : $+$, c'est-à-dire :

$$\forall A, B \in G \quad \text{on a :} \quad A + B \in G$$

avec

$$A + B = \{U \mid U \in A \text{ ou } U \in B\}$$

Produit ou concaténation : que l'on notera ; .

$$\forall A, B \in G \quad \text{on a} \quad A \,.\, B \in G$$

avec

$$A \,.\, B = \{U \mid \exists\, V \in A \text{ et } W \in B \text{ tels que } U = V \,.\, W$$

Eléments neutres :

Il existe un élément neutre ϕ pour la somme et un élément neutre $\mathfrak{J}$ pour le produit, ces éléments appartenant à G, tels que :

pour tout $A \in G$, on ait :

$$\boxed{\begin{array}{l} A + \phi = \phi + A = A \\ A \,.\, \mathfrak{J} = \mathfrak{J} \,.\, A = A \end{array}}$$

ϕ : est l'ensemble qui ne contient aucune partie du monoïde M, appelée "partie vide".

$\mathfrak{J} = \{I\}$; est la partie réduite ou mot vide.

Propriétés :

Associativité de la somme et du produit :

$$\boxed{\begin{array}{l} \forall A, B, C \in G \qquad \text{on a :} \\ A + (B + C) = (A + B) + C \\ A \,.\, (B \,.\, C) = (A \,.\, B) \,.\, C \end{array}}$$

Commutativité de la somme

$$\boxed{\begin{array}{l} \forall A, B \in G \qquad \text{on a :} \\ A + B = B + A \end{array}}$$

Remarque :

En général le produit n'est pas commutatif.

Distributivité du produit par rapport à la somme :

$$\boxed{\begin{array}{l} \forall A, B, C \in G \qquad \text{on a :} \\ A \cdot (B + C) = A \cdot B + A \cdot C \\ (A + B) \cdot C = A \cdot C + B \cdot C \end{array}}$$

Nota

La somme n'est pas distributive par rapport au produit.

Idempotence de la somme

$$\boxed{\forall A \in G \quad \text{on a} \quad A + A = A}$$

Convention relative à la partie vide

$$\boxed{\forall A \in G \quad \text{on pose} \quad A \cdot \phi = \phi \cdot A = \phi}$$

Opération itération ou "étoile"

Conventions :

Dans les cas où tous les facteurs d'un produit sont identiques : On pose :

$$\boxed{\begin{array}{l} A^1 = A \\ A^n = A^{n-1} \cdot A \end{array}}$$

Conformément à la définition du produit, A^n est l'ensemble des mots formés par juxtaposition de n mots appartenant à A, ces mots n'étant pas nécessairement identiques, dans le cas où A contient plus d'un mot.

On peut aussi poser, par convention :

$$\boxed{A^\circ = \mathcal{J}}$$

L'itération de $A \in G$ notée A^* , est l'ensemble :

$$\boxed{A^* = \mathcal{J} + A + A^2 + A^3 + \ldots}$$

Comme $\mathcal{J} = A^\circ$, on peut écrire :

$$\boxed{A^* = \sum_{i=o}^{\infty} A^i}$$

A^* est l'ensemble de tous les mots formés par juxtaposition de mots de A .

Remarques

1 - Si A ne possède qu'un seul générateur libre x , A^* est appelé "ensemble cyclique infini".

2 - Si A est l'ensemble des éléments libres, soit X , le monoïde libre relatif à X est :

$$M = X^*$$

d'où le nom de système de générateurs de M donné à X [cf : 2]

Opération itération définie

L'itération définie de $A \in G$ notée A^+ , est l'ensemble :

$$\boxed{A^+ = A + A^2 + A^3 + \ldots}$$

que l'on peut écrire :

$$\boxed{A^+ = \sum_{i=1}^{\infty} A^i = A \,.\, A^*}$$

d'où $\boxed{A^* = \mathcal{I} + A^+}$

Nota

Si A ne contient pas I , A^k non plus, donc A^+ ne contient pas I non plus.

Si $I \in A$, on peut écrire :

$$A^+ = A^* = \lim_{n \to \infty} A^n = A^\infty$$

Propriétés de l'itération

On vérifie les propriétés suivantes :

$$\mathcal{I}^* = \mathcal{I}$$

$$\phi^* = \phi$$

Idempotence de l'itération : $(A^*)^* = A^*$

Loi d'absorption : $A^* \,.\, A^* = A^*$

$$A^* + A^n = A^*$$

$$A^* \,.\, A + \mathcal{I} = A \,.\, A^* + \mathcal{I} = A^*$$

$$(A + B)^* = (A^* \,.\, B^*)^*$$

$$(A + B)^* = (A^* \,.\, B)^* \,.\, A^*$$

$$(A \,.\, B)^* A = A \,.\, (B \,.\, A)^*$$

Autres propriétés (utilisées dans la description d'ALGOL)

$$\boxed{A^* + A \,.\, A^* = A^*}$$

Car : $A \,.\, A^* = A^+$ et $A^* = A^+ + \mathcal{I}$

d'où $A^+ + \mathcal{I} + A^+ = A^+ + A^+ + \mathcal{I}$

Comme : $A^+ + A^+ = A^+$

On a : $A^+ + A^+ + \mathcal{J} = A^*$

On a aussi :

$$\boxed{\begin{aligned} A^* + A \,.\, A^* &= (\mathcal{J} + A) \,.\, A^* \\ &= A^* + A^+ \end{aligned}}$$

<u>Remarque</u> :

L'ensemble $\mathcal{P}(M)$ est un "<u>Gerbier libre</u>" :

On appelle ainsi une structure algébrique à deux opérations :

- une addition associative et commutative, à élément neutre, et idempotente
- une opération de "<u>concaténation</u>" , associative et distributive par rapport à la somme, à élément neutre.

L'étude du Gerbier libre permet celle des séquences d'objets ou d'abstractions quelconques.

L'algèbre du Gerbier libre s'appliquera donc à l'étude :

- des graphes
- des réseaux logiques séquentiels
- des automates
- des langages de programmation
- etc ...

VARIABLES, EXPRESSIONS, FONCTIONS, EQUATIONS DANS L'ALGEBRE DU GERBIER LIBRE

Variables libres

Ce sont des variables dont les valeurs appartiennent à un gerbier libre.

Expressions libres

On appelle expression libre, toute expression algébrique formée à partir de variables libres et de constantes appartenant à un gerbier libre, par un nombre fini d'applications des opérations + , . et * .

Pour définir l'ordre dans lequel les opérations doivent être exécutées, on est amené à employer des parenthèses : afin de permettre un allègement d'écriture, on a défini l'ordre de priorité suivant entre opérateurs (ordre décroissant) :

1 - *
2 - .
3 - +

Fonction engendrée par une expression libre

Si l'on remplace les variables d'une expression libre $F(X_1, X_2 \ldots X_n)$ par des éléments particuliers du gerbier libre G, on obtient un résultat qui est un élément de G.

Si l'on donne aux variables toutes les combinaisons de valeurs possibles, et on calcule pour chacune la valeur de l'expression, on voit que par l'intermédiaire de l'expression on définit une fonction $F(X_1, X_2 \ldots X_n)$ de G^n dans G.

Avec une expression d'une variable libre, on définirait une fonction de G dans G.

Equations

Etant données deux fonctions libres f et g de G dans G.
Un élément X_o de G tel que :

$$f(X_o) = g(X_o)$$

s'appelle solution de l'équation :

$$f(X) = g(X)$$

Simplification des expressions

Comme en algèbre habituelle, on peut transformer les expressions libres au moyen des divers axiomes et théorèmes de l'algèbre du gerbier libre.

Exemples de variables, expressions, fonctions, équations

Soient : $N = \{0, 1\}$: système de générateurs libres
$M = N^*$: monoïde libre relatif à N
$G = \mathcal{P}(M) = \mathcal{P}(N^*)$: gerbier libre

et soit $X \in G$

X est une expression

$X . 1$ et $X . 0$ sont des expressions

$Y = X . 0$ est une fonction qui à tous les éléments de G fait correspondre tous les éléments de la forme Y.
Leur ensemble est l'ensemble des représentations binaires des nombres pairs.

$X = (0 + 1) \,.\, X + 1$ est une équation, dont la solution est l'ensemble des représentations binaires de tous les nombres impairs.

<u>Résolution de l'équation</u>

$$(1) \quad \boxed{X = X \,.\, A + B}$$

On a, en multipliant les deux nombres par A :

$$X \,.\, A = X \,.\, A^2 + B \,.\, A$$

en remplaçant $X \,.\, A$ par son expression dans (1), on a :

$$X = X \,.\, A^2 + B \,.\, A + B$$

en continuant comme ci-dessus, on obtient :

$$X = X \,.\, A^n + B \,.\, (\mathfrak{J} + A + A^2 + \ldots + A^{n-1})$$

soit $f \in X$ de longueur k

Si $n > k$ et si $\mathfrak{J} \notin A$, $f \notin X \,.\, A^n$ qui ne contient que des mots de longueur $\geqslant n$

donc $f \in B \,.\, (\mathfrak{J} + A + A^2 + \ldots + A^{n-1}) \subset B \,.\, A^*$

On a donc : $X \subset B \,.\, A^*$

Réciproquement, si $x \in B \,.\, A^*$
$\exists\, k$ tel que $x \in B \,.\, A^k$

Or, pour $n > k$ $B \,.\, A^k \subset X$ donc $x \in X$ et $B \,.\, A^* \subset X$

On a donc

$$\boxed{\begin{array}{c} \text{si } \mathfrak{J} \notin A \\ \text{la solution de } X = X \,.\, A + B \\ \text{est } X = B \,.\, A^* \end{array}}$$

Si $\mathcal{J} \in A$

On a : $X = X \,.\, \mathcal{J} \subset X \,.\, A \subset X \,.\, A + B$

soit $C \supset B$

C quelconque et $X = C \,.\, A^*$

On a : $X \supset C \,.\, A^* \supset B$ et $X = X \,.\, A$

donc : $X \supset X \,.\, A + B$ et $X = X.A + B$

Il y a donc une infinité de solutions dans ce cas.

<u>Nota</u>

On démontrerai de même que la solution de $X = A \,.\, X + B$ est :

$$X = A^* \,.\, B \quad \text{si} \quad \mathcal{J} \notin A$$

<u>Application à l'exemple précédent</u>

Soit $X = (0 + 1) \,.\, X + 1$

Comme $\mathcal{J} \notin (0 + 1)$

on a :

$$\boxed{X = (0 + 1)^* \,.\, 1}$$

LANGAGES DE PROGRAMMATION

Définition : [cf : 4 et 7]

On appelle langage de programmation L , un sous-ensemble du monoïde libre X^* relatif à un système de générateurs libres X , non vide, appelé ensemble de base ou vocabulaire terminal.

Si X : est l'ensemble de base

X^* : monoïde libre relatif à X

$L \subseteq X^*$ est un langage défini sur X

L peut être défini par une équation dans l'algèbre du gerbier libre $G = \mathcal{P}(X^*)$

L est alors la solution de l'équation définie ci-dessus.

On utilise en général un système d'équations dont la solution est un ensemble de langages dont L fait partie.

Exemples de langages de programmation

Langages de Chomsky

Soient $X = \{a_1, a_2 \dots a_n\}$ l'ensemble de base

$G = \mathcal{P}(X^*) = \{L_1, L_2, L_3 \dots\}$ l'ensemble des parties du monoïde libre relatif à X

$L_1 \in G$, $L_1 \subset X^*$ est un langage de Chomsky s'il est défini à partir d'un système d'équations de la forme :

$$L_j = f_j\ (a_i, L_i)$$

$$a_i \in X \qquad L_i, L_j \in G$$

1er exemple

Soient $X = \{a_1, a_2, a_3, a_4\}$

$G = \{L_1, L_2, L_3 \ldots\}$

Le système

$$\begin{cases} L_1 = L_2 \cdot L_3 \\ L_2 = L_1 \cdot a_3 + a_1 \\ L_3 = a_4 \cdot L_3 + a_2 \end{cases}$$

permet d'obtenir le langage L_1 , qui est un langage de Chomsky

2ème exemple

Soient $X = \{a, b\}$

$G = \{L_1, \ldots\}$

La solution de :

$$L_1 = a \cdot L_1 \cdot b + a \cdot b$$

est un langage de Chomsky

3ème exemple

Soient $X = \{a_1, a_2, a_3\}$

$G = \{L_1, \ldots\}$

La solution de

$$L_1 = a_1 \cdot L_1 \cdot a_1 + a_2 \cdot L_1 \cdot a_2 + a_3$$

est un langage de Chomsky

Langages réguliers de Kleene

Soient $X = \{a_1, a_2 \dots a_n\}$ l'ensemblede base

$G = \mathcal{P}(X^*) = \{L_1, L_2, L_3, \dots\}$ l'ensemble des parties du monoïde relatif à X .

$L_1 \in G$, $L_1 \subset X^*$ est un langage régulier de Kleene s'il est défini à partir d'un système d'équations de la même forme qu'un système de Chomsky, mais où la partie droite de chaque équation est une somme de termes de la forme :

soit : $a_j \cdot L_k$

soit : $L_k \cdot a_j$

soit : a_j

Exemples

Soient $X = \{a_1, a_2\}$

$G = \{L_1, L_2 \dots\}$

Le système

$$\begin{cases} L_1 = a_1 \cdot L_1 + a_2 \cdot L_2 \\ L_2 = a_2 \cdot L_2 + a_2 \end{cases}$$

Permet d'obtenir le langage L_1 qui est un langage régulier de Kleene.

Langage machine

Considérons un calculateur dont les instructions sont à deux adresses, dont une est constante correspondante à un registre mémoire particulier.

Chaque instruction Is est formée en général, sans tenir compte de cas particuliers de :

- un code opération (machine) : Op
- une adresse variable : Ad

soit

Is = Op . Ad

Un calculateur possède en général, un nombre fini de codes opération et un nombre fini d'adresses.

Considérons l'ensemble :

$$X = \{Op_1, Op_2, \ldots Op_n, Ad_1, Ad_2, \ldots Ad_m\}$$

soit $G = \mathcal{P}(X^*) = \{Is, P, U, V, \ldots\}$

avec $U = \{Op_1, Op_2, \ldots Op_m\}$

$V = \{Ad_1, Ad_2, \ldots Ad_m\}$

L'ensemble des instructions est :

$$\boxed{Is = U \,.\, V}$$

C'est l'ensemble des mots formés par juxtaposition d'un mot de U avec un mot de V.

Un programme étant une suite d'instructions, non vide, il peut être défini par :

$$P = P \,.\, Is + Is$$

ou $P = (Is)^+$

soit $\boxed{P = (U \,.\, V)^+}$

<u>Langage machine à deux adresses</u>

Reprenant les ensembles ci-dessus

$$X = \{Op_1, Op_2, \ldots, Op_n, Ad_1, Ad_2, \ldots Ad_n\}$$

$$G = \mathcal{P}(X^*) = \{Is, P, U, V, \ldots\}$$

Une instruction à deux adresses est de la forme :

$$Is = Op \,.\, Ad_i \,.\, Ad_j$$

L'ensemble des instructions est

$$\boxed{Is = U \,.\, V^2}$$

Un programme est défini par :

$$\boxed{P = (U \,.\, V^2)^+}$$

DEUXIEME PARTIE

RECONNAISSANCE D'UN LANGAGE

AUTOMATE DE RECONNAISSANCE D'UN LANGAGE

Définition

Etant donné un ensemble de base X , un langage $L \subseteq X^*$ et un mot $w \in X^*$, on appelle automate de reconnaissance de L , un système qui fait correspondre à w , une réponse R susceptible de prendre deux valeurs (1 ou 0, vrai ou faux, oui ou non, etc ...) , par exemple :

$$R = 1 \quad \text{si} \quad w \in L$$
$$R = 0 \quad \text{si} \quad w \notin L$$

L est le langage reconnu par l'automate, c'est l'ensemble des w pour lesquels $R = 1$.

Description d'un automate de reconnaissance associé à un système d'équations

Dans tous les schémas et descriptions qui suivent, on se réfère toujours à :

$X = \{a, b, c, d, a_i, x_i, \ldots, n\}$ système de générateurs libres

$X^* = \{w, w_i, u_i, v_i, \ldots\}$ monoïde libre relatif à X

Les éléments libres sont considérés comme des mots à une seule lettre, c'est-à-dire qu'on plonge X dans X^*

Un mot de X^* a la forme générale :

$$(1) \qquad w = x_1 \; x_2 \; \ldots \; x_n$$

et peut s'écrire aussi

$$(2) \qquad w = w_1 \cdot w_2 \ldots w_m$$

$w_1, w_2, \ldots, w_m$ sont appelés sous-mots du mot (1)

$$G = \mathcal{P}(X^*) = \{A, B, C, L, \ldots\}$$

ensemble des parties du monoïde relatif à X .

Principe d'un automate de reconnaissance :

Le principe général de l'automate de reconnaissance construit ici est le suivant :

Etant donnés : $L = \{w_1, w_2, \ldots\}$

et $w \in X^*$

l'automate doit vérifier $w \in L$, sinon l'automate signale qu'il y a une erreur dans l'écriture de w.

Vérifier que $w \in L$, c'est vérifier que :

$w = w_1$ ou $w = w_2$ etc ...

De plus, vérifier que :

$w = w_1$

1) Dans le cas le plus simple, si $w_1 = a_1 a_2 \ldots a_n$, c'est vérifier que w est de longueur n, c'est-à-dire que w est de la forme : $w = x_1 x_2 \ldots x_n$ et que $x_1 = a_1$ et $x_2 = a_2 \ldots$ et $x_n = a_n$.

2) Dans le cas général où $w_1 = u_1 . u_2 \ldots u_p$ ($u_1, u_2, \ldots u_p$: sous-mots de w_1), c'est vérifier que w est de la forme :

$$w = v_1 . v_2 \ldots v_p$$

et que $v_1 = u_1$ et $v_2 = u_2 \ldots$ et $v_p = u_p$

On est donc ramené au cas le plus simple (reconnaissance d'éléments de base)

Toutes ces vérifications sont effectuées par différents sous-programmes, reliés entre eux suivant un ordre déterminé par l'équation ou le système d'équations définissant L.

La construction de l'automate de reconnaissance va dépendre de la forme de l'équation ou du système d'équations définissant L.

Cependant, on peut apporter une simplification de l'automate en transformant le système d'équations permettant d'obtenir L, en un système où chaque équation a la forme :

$$L = a \cdot L_2 + L_3$$

et dans laquelle l'expression de L_3 ne commence pas par a.

Ceci peut être obtenu :

1 - en effectuant tous les développements dans les expressions du système d'équations

2 - en remplaçant éventuellement certaines variables par leurs expressions de façon à faire apparaître des facteurs gauche communs.

3 - en effectuant les mises en facteur gauche dans les expressions et ce jusqu'à ce que toutes les équations soient de la forme définie ci-dessus.

L'utilisation d'un tel système d'équations, permettra de construire un automate simple et rapide, à fonctionnement séquentiel. Celui-ci analysera un mot élément par élément. Chaque fois qu'un élément est reconnu correct, celui-ci peut être éliminé de l'analyse et l'automate n'analyse que la suite du mot.

En reprenant l'équation :

$$L = a \cdot L_2 + L_3$$

avec l'expression de L_3 ne commençant pas par a.

Vérifier que $w \in L$, c'est vérifier que

1 - w est de la forme $x \cdot U_1$ et que $x = a$ et $u_1 \in L_2$

ou

2 - $w \in L_3$

Si $x \neq a$, alors si $x \cdot U_1 \in L_3$, $w \in L$

Si $x \neq a$, et si $x \cdot U_1 \notin L_3$, $w \notin L$

Mais si $x = a$, alors $x \cdot U_1 \notin L_3$

et si $u_1 \notin L_2$, alors $w \notin L$

L'automate signale une erreur dans les cas :

$x \neq a$ et $x \cdot U_1 \notin L_3$

ou

$x = a$ et $U_1 \notin L_2$

Une autre simplification peut être apportée au système d'équations.

Dans le cas où l'on ne peut plus faire de nouvelle mise en facteur à gauche, alors on peut faire des mises en facteur à droite.

<u>Remarque importante</u>

On suppose que le langage que l'on analyse ne peut pas être vide.

Autrement dit on ne peut pas trouver pour expression du langage L :

$$L = L_1 + \mathfrak{J}$$

c'est-à-dire que $\mathfrak{J}$ n'interviendra que :

soit à droite

soit à gauche

soit au milieu

d'un mot de L .

Exemple de transformation d'un système

Soit le système d'équations permettant d'obtenir L :

$$\text{I} \quad \begin{cases} L = a \,.\, d \,.\, L + b \,.\, L_1 + L_2 & (1) \\ L_1 = d \,.\, L_1 + e & (2) \\ L_2 = (a \,.\, L_1 + d) \,.\, b \,.\, c & (3) \end{cases}$$

Après développement de (3) :

$$L_2 = a \,.\, L_1 \,.\, b \,.\, c + b \,.\, c$$

puis après remplacement de L_2 dans (1) , on a :

$$\text{II} \quad \begin{cases} L = a \,.\, d \,.\, L + b \,.\, L_1 + a \,.\, L_1 \,.\, b \,.\, c + b \,.\, c \\ L_1 = d \,.\, L_1 + e \end{cases}$$

en effectuant les mises en facteurs gauche, on obtient :

$$\text{III} \quad \begin{cases} L = a \,.\, (d \,.\, L + L_1 \,.\, b \,.\, c) + b \,.\, (L_1 + c) \\ L_1 = d \,.\, L_1 + e \end{cases}$$

que l'on peut écrire :

$$\text{IV} \quad \begin{cases} L = a \,.\, L'_1 + L'_2 & (1') \\ L'_1 = d \,.\, L + L_1 \,.\, b \,.\, c & (2') \\ L'_2 = b \,.\, (L_1 + c) & (3') \\ L_1 = d \,.\, L_1 + e & (4') \end{cases}$$

En recommençant la transformation sur (2') et (4') , on a :

$$L'_1 = d \,.\, L + (d \,.\, L_1 + e) \,.\, b \,.\, c$$

soit : $L'_1 = d \,.\, L + d \,.\, L_1 \,.\, b \,.\, c + e \,.\, b \,.\, c$

on obtient alors, en remplaçant L'_2 par son expression :

$$\text{V} \quad \begin{cases} L = a \,.\, L'_1 + b \,.\, (L_1 + c) \\ L'_1 = d \,.\, (L + L_1 \,.\, b \,.\, c) + e \,.\, b \,.\, c \\ L_1 = d \,.\, L_1 + e \end{cases}$$

Fonctionnement d'un automate de reconnaissance

Un mot $w \in X^*$ se présente sous la forme d'une suite d'éléments libres.

Il faut bien entendu indiquer le commencement et la fin du mot par un caractère spécial que l'on note ici ##

Cette suite d'éléments de base est perforée sur un ruban papier (PB 250).

L'automate reconnaît le caractère spécial ## de début de mot, puis commence la reconnaissance de w . Celle-ci s'effectue élément par élément.

Tant qu'un élément n'a pas pu être reconnu correct, la suite des éléments libres à analyser n'est pas modifiée. Quand un élément est reconnu bon, celui-ci est éliminé de la suite et l'automate doit analyser l'élément suivant. Quand tout le mot est reconnu correct, l'automate vérifie s'il trouve bien le caractère spécial ## indiquant la fin du mot.

Construction d'un automate de reconnaissance

1°) Elément d'automate associé à un élément libre

Appelons "élément d'automate" le système qui fait correspondre à un élément libre $x_i \in X$ une réponse R susceptible de prendre deux valeurs (0 ou 1 , non ou oui, etc ...) , par exemple :

soit $a \in X$

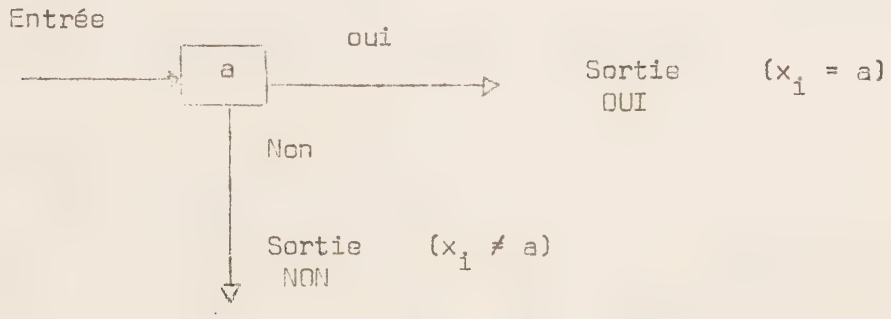
$R = 1$ si $x_i = a$

$R = 0$ si $x_i \neq a$

De plus, si la réponse est $R = 1$, l'élément d'automate élimine l'élément libre x_i .

Dans un calculateur arithmétique, on réalise l'élément d'automate par l'intermédiaire d'un sous-programme capable de vérifier : $x_i = a$ (opération très simple pour le calculateur, par comparaison des représentations machine de x_i et de a).

Ce sous-programme comporte une entrée et deux sorties, et peut-être représenté par le schéma :



A l'intérieur du quadrilatère, on inscrit simplement le nom a étant sous-entendu que le sous-programme vérifie $x_i = a$ et qu'en cas de réponse affirmative, il élimine x_i .

L'appel d'un tel sous-programme se fait de façon classique, mais avec deux instructions de retour.

Exemple d'utilisation

Si l'élément d'automate (a) doit analyser le premier élément libre du mot a b c ##, la sortie s'effectue par oui et il reste à analyser b c ##

Si le mot était b c d ##, la sortie s'effectuerait par non et il resterait à analyser : b c d ##

2°) Partie d'automate associée à une somme :

Soit : $\boxed{L = A + B}$

avec : $A = \{a\}$ et $B = \{b\}$

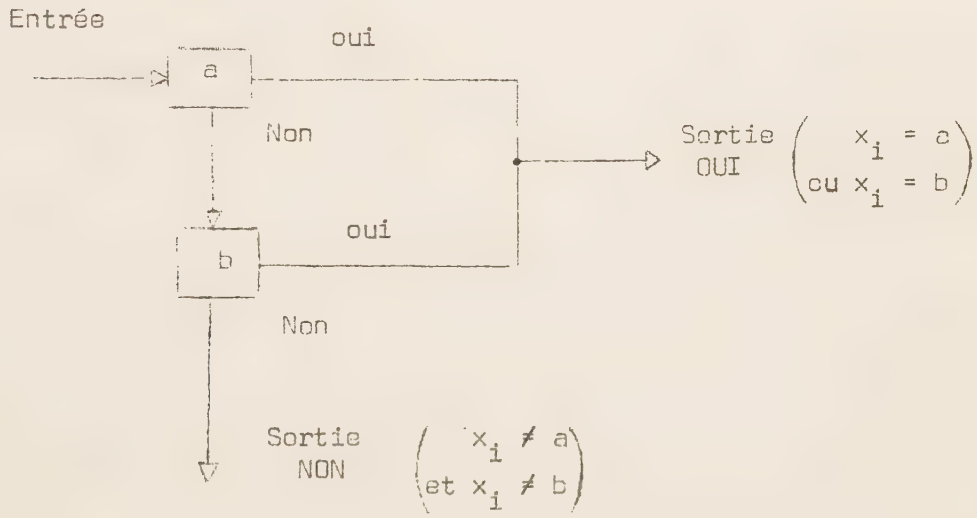
L comprend deux éléments libres $L = \{a, b\}$ et soit $x_i \in X$.

La partie d'automate, associée à L , vérifie si :

$x_i = a$ ou $x_i = b$, c'est-à-dire $x_i \in L$

Si la réponse est oui , celle-ci élimine x_i .

On utilise deux éléments d'automate reliés suivant le schéma :



Exemple d'utilisation

Si la partie d'automate (A + B) doit analyser le premier élément libre du mot a b c ##, la sortie s'effectue par oui et il reste à analyser b c ##

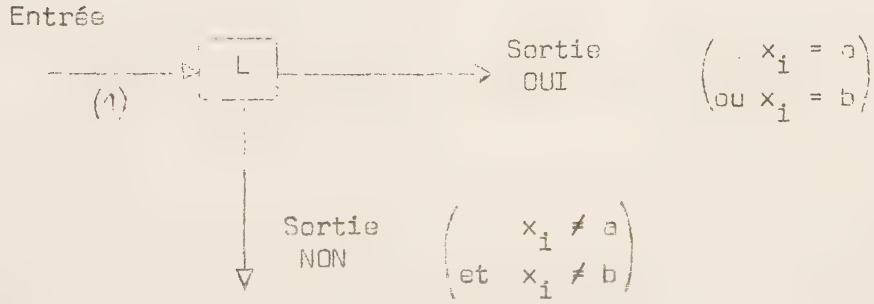
Si le mot était b c d ##, la sortie s'effectuerait aussi par oui et il resterait à analyser c d ##.

Si le mot était c d e ##, la sortie s'effectuerait par non et il resterait à analyser c d e ##

Remarques

1 - On aurait pu construire la partie d'automate "Somme" correspondante à (B + A) équivalent à (A + B)

2 - Pour des raisons de simplification, on peut représenter la partie d'automate "Somme" par le schéma



avec $L = A + B$ $\quad L = \{a, b\}$

$A = \{a\}$ et $B = \{b\}$

On peut dire que (1) est la partie d'automate qui vérifie

$x_i \in L$

3°) Partie d'automate associée à un produit

soit $\boxed{L = A \cdot B}$

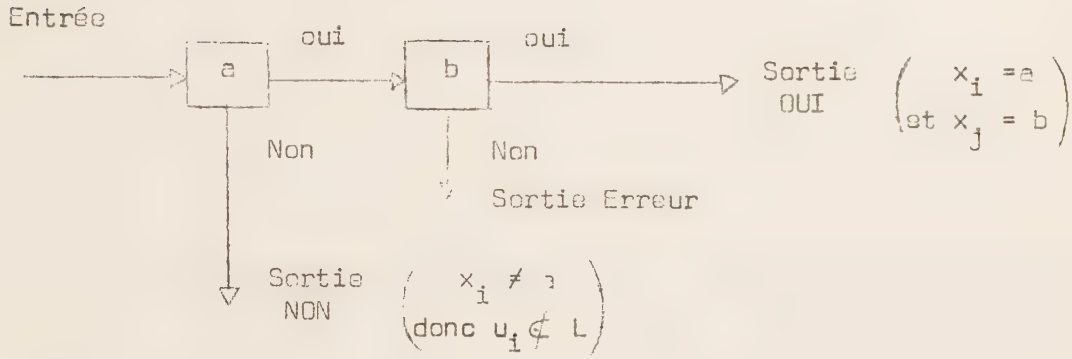
avec : $A = \{a\}$ et $B = \{b\}$

L comprend un seul mot : $L = \{a. b\}$ et soit un sous-mot $U_i = x_i \cdot x_j \qquad x_i , x_j \in X$

La partie d'automate associée à L , vérifie si $U_i = a \cdot b$ soit $x_i = a$ et $x_j = b$, c'est-à-dire $U_i \in L$

Si la réponse est oui, celle-ci élimine $x_i \, x_j$.

On utilise deux éléments d'automates reliés suivant le schéma :



Sortie Erreur :(x_i = a et $x_j \neq$ b , donc $U_i \notin L$ mais aussi ne pourra pas être reconnu par aucun autre élément d'automate faisant partie de l'automate complet à fonctionnement séquentiel puisqu'on a mis a en facteur)

Exemple d'utilisation

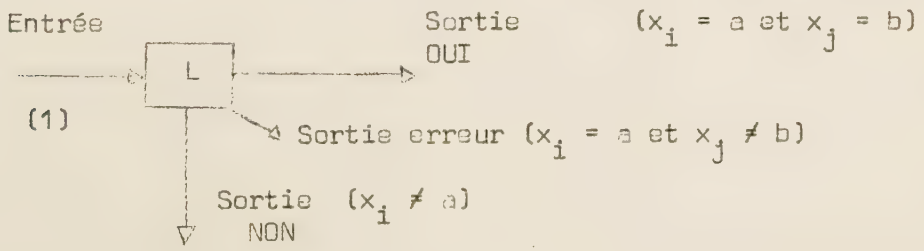
Si la partie d'automate (A . B) doit analyser le premier sous-mot de longueur 2 du mot a b c d ## , la sortie s'effectue par oui et il reste à analyser c d ##

Si le mot était b c d e ## , la sortie s'effectuerait par Non et il resterait à analyser b c d e ##

Si le mot était a c d e ## , la sortie s'effectuerait par Erreur.

Remarque

Pour des raisons de simplification, on peut représenter la partie d'automate "produit" par le schéma :



avec : $L = A \cdot B$ $\quad L = \{a \cdot b\}$

$A = \{a\}$ et $B = \{b\}$

On peut dire que (1) est la partie d'automate qui vérifie

$$x_i \cdot x_j \in L$$

4°) Partie d'automate associée à une expression libre quelconque

On peut associer à une expression libre quelconque une partie d'automate. La construction de celle-ci est effectuée à partir de l'expression transformée sous forme séquentielle, en effectuant toutes les mises en facteur à gauche et en utilisant les schémas élémentaires "Somme" et "Produit"

Exemple

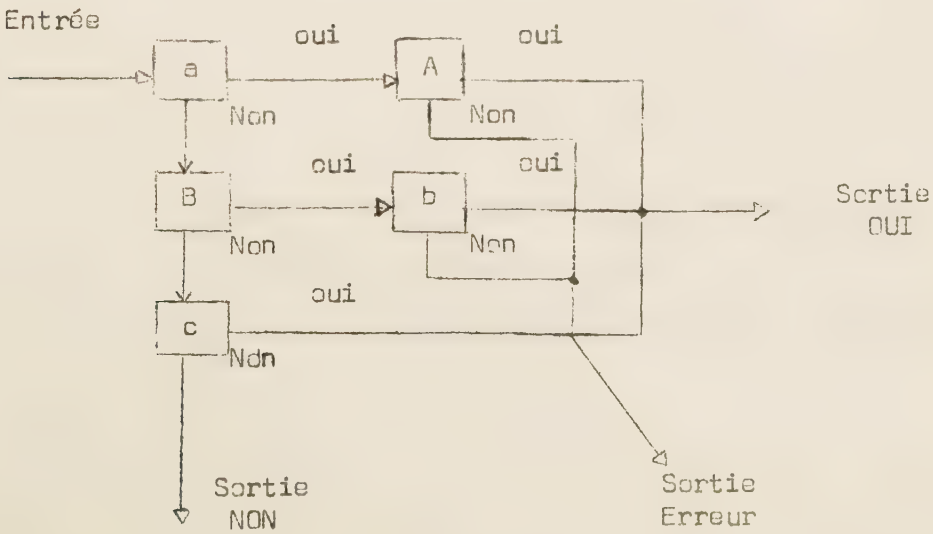
Soit $\boxed{L = a \cdot A + B \cdot b + c}$

avec $a, b, c \in X$

$A, B, L \in G$

et l'expression de B ne commençant pas par a.

La partie d'automate associée à L peut être représentée par le schéma :



5°) Cas particulier du vide

a) Soit l'expression : $L = A \cdot \mathfrak{J}$

On sait que : $L = A$

donc il y a équivalence des schémas ci-dessous :

(1)

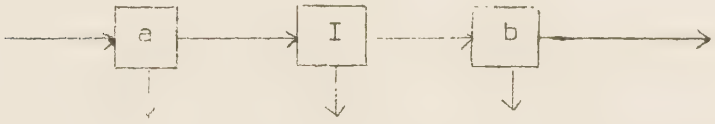
(2)

Le premier de ces schémas doit être utilisé si l'on écrit le symbole neutre I .

En général, celui-ci n'est pas écrit et l'on utilise le deuxième schéma.

De même, il y a équivalence entre les schémas ci-dessous :

(1)



(2)

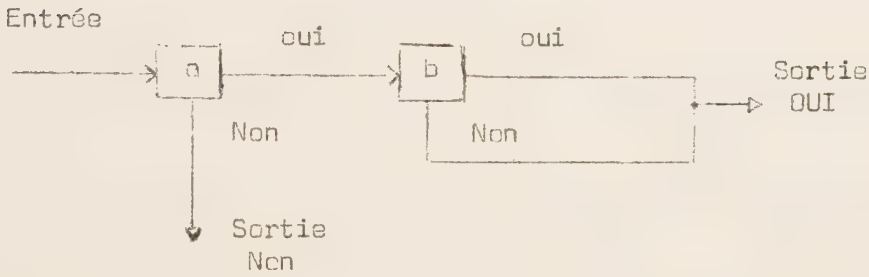
Par convention, on n'écrira pas le symbole neutre I à l'intérieur d'un mot, celui-ci étant un symbole qui peut être omis.

On utilisera donc systématiquement les schémas du type ci-dessus (2).

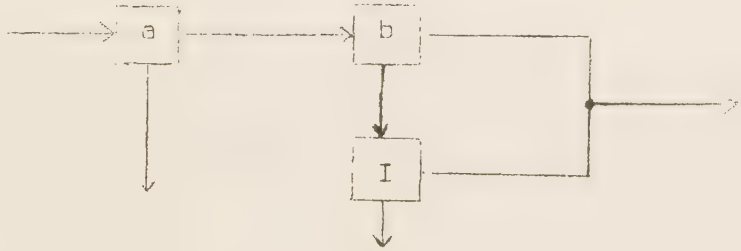
b) Application

Soit l'expression : $L = A \cdot (B + \bar{J})$

On obtient le schéma :



équivalent du schéma :



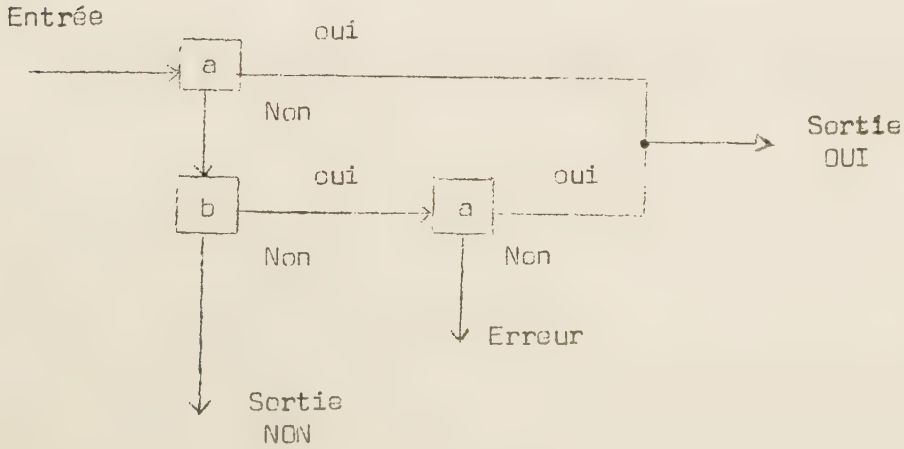
Remarque

Soit $L = (\bar{J} + B) \cdot A$

On doit développer

d'où $L = A + B \cdot A$

d'où le schéma de la partie d'automate :

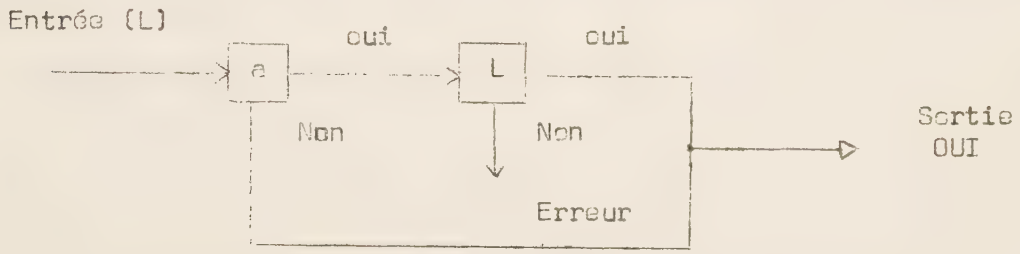


6°) Partie d'automate associée à une opération étoile

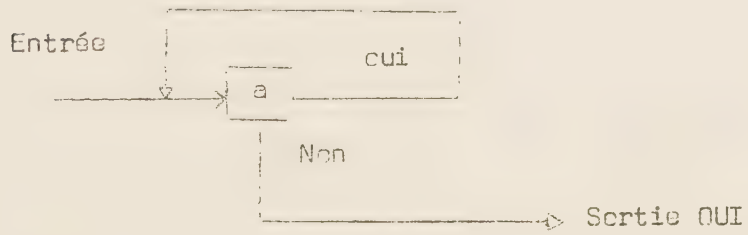
soit $L = a^*$

ou $L = a \cdot L + \emptyset$

La partie d'automate associée à L peut être représentée par le schéma :



Comme la sortie erreur de L n'existe pas, la sortie de (a) peut être reliée directement à l'entrée (L) et l'on obtient le schéma :



7°) Partie d'automate associée à une opération itération définie

soit $L = a^+$

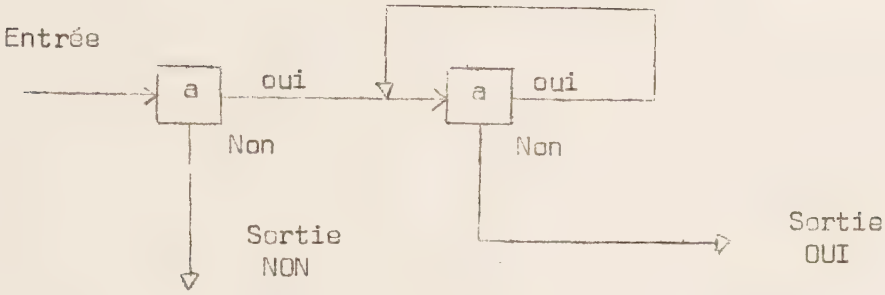
ou : $L = a + a^2 + a^3 + \ldots$

En mettant a en facteur à gauche, on a :

$$L = a \cdot (I + a + a^2 + \ldots)$$

$$L = a \cdot a^*$$

La partie d'automate associée à (a^+) peut être représentée par le schéma :

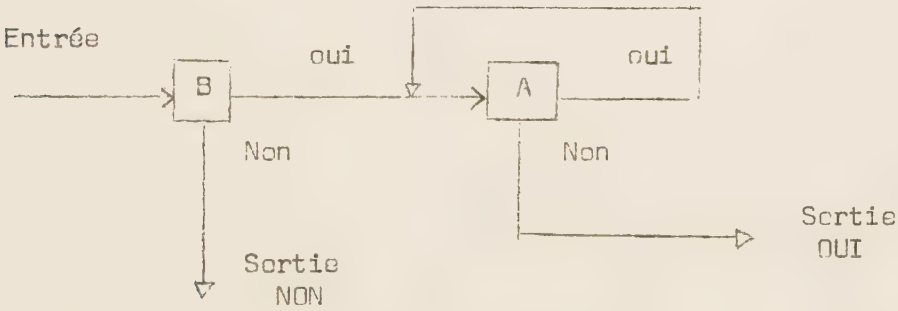


<u>Autres exemples de parties d'automates types utilisés pour la reconnaissanc du langage ALGOL 60</u>

1°) Soit $\boxed{L = L \,.\, A + B}$

ou $L = B \,.\, A^*$ si $I \notin A$

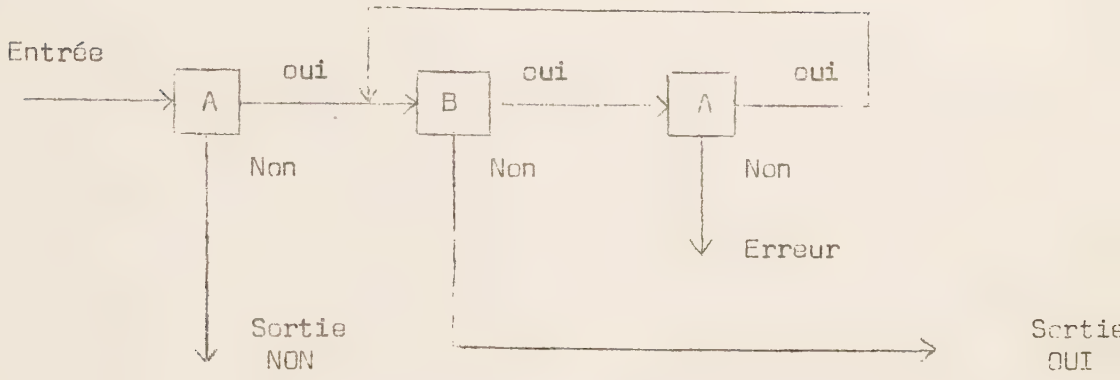
On obtient le schéma de la partie d'automate :



2°) Soit $\boxed{L = L \,.\, B \,.\, A + A}$

ou $L = A \,.\, (B \,.\, A)^*$

On obtient le schéma de la partie d'automate

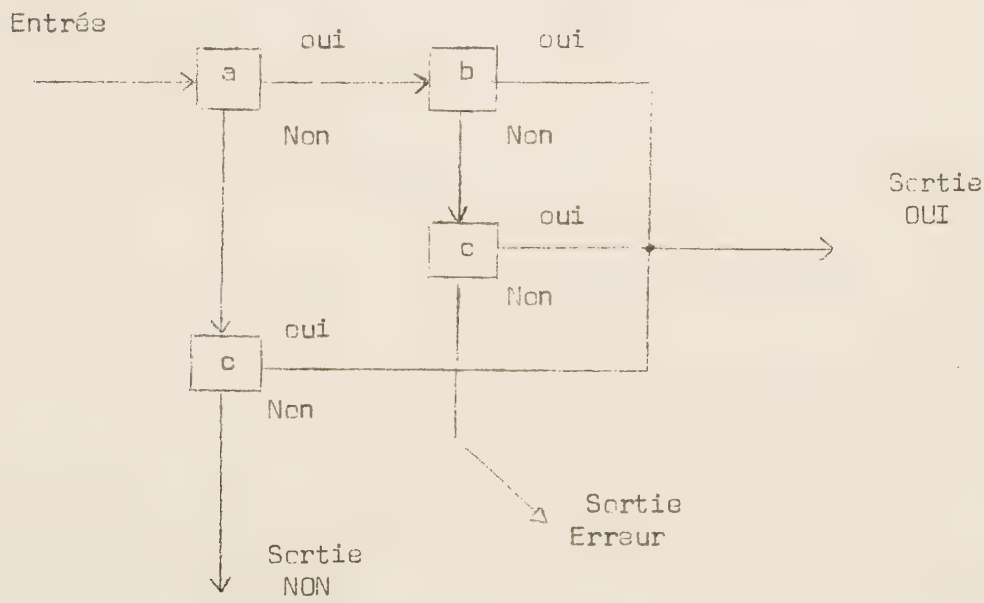


3°) Soit $\boxed{L = A \,.\, B + A \,.\, C + C}$

On doit faire la mise en facteur à gauche

d'où $L = A \,.\, (B + C) + C$

on obtient le schéma de la partie d'automate



Automate associé à un système d'équations permettant d'obtenir un langage L

Le système d'équations est supposé être transformé sous forme séquentielle, en effectuant toutes les mises en facteur à gauche.

L'automate associé à un système d'équations est un système de parties d'automate.

Chaque partie d'automate correspond à une expression, membre de droite d'une équation.

Ce système de parties d'automate est complété par deux sous-programmes de contrôle des caractères // de début et de fin de mot.

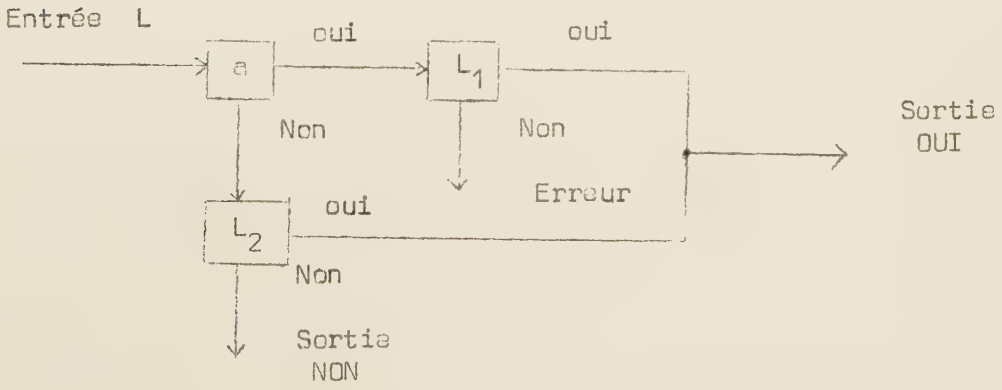
L'appel de ces parties d'automate correspondant à chaque équation se fait par l'intermédiaire d'une pile de mémoires. C'est ce que l'on appelle "Automate à piles".

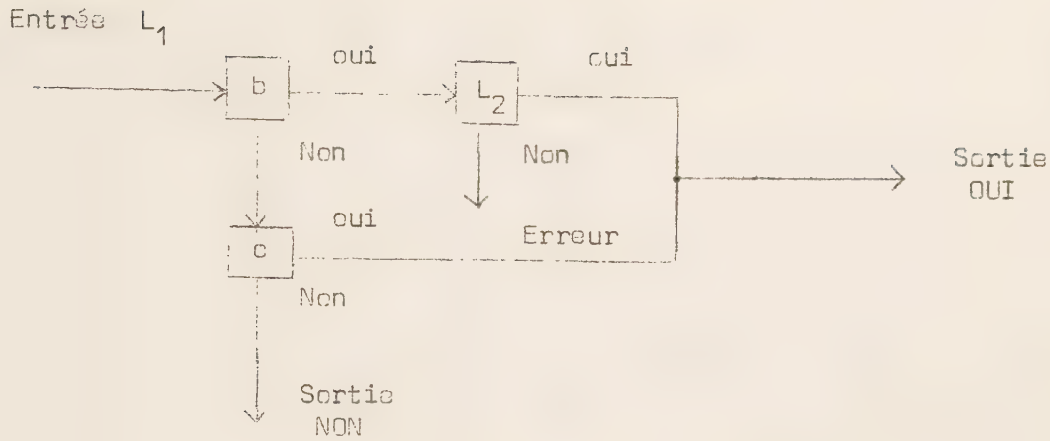
A l'appel de chaque partie d'automate, les deux instructions de retour sont empilées et à la sortie, ce sont les deux dernières instructions de retour qui sont utilisées, détruites et remplacées éventuellement par les deux instructions de retour suivantes.

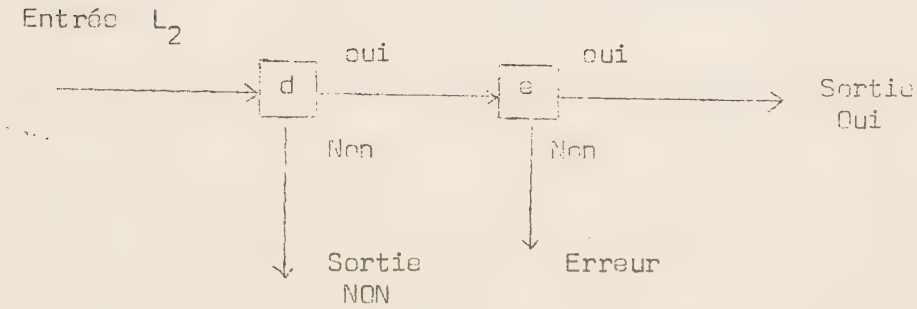
Exemple d'automate

Soit le système d'équations

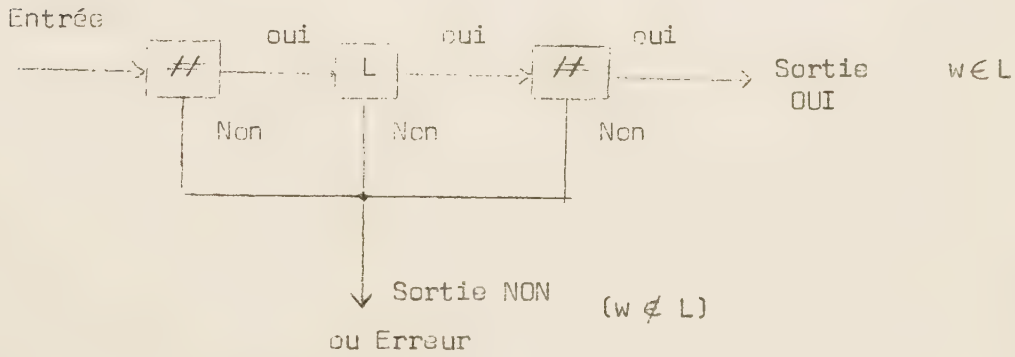
$$\begin{cases} L = a \cdot L_1 + L_2 \\ L_1 = b \cdot L_2 + c \\ L_2 = d \cdot e \end{cases}$$







L'automate est représenté par le schéma



on peut illustrer le fonctionnement par un exemple.

Soit $w \in X^*$ représenté par # a b d e #

L'automate reconnait le caractère ## , alors a b d e ## va être analysé par (L)

L'élément (a) de (L) va reconnaître le premier élément libre de w , il restera après le passage dans (a) , à analyser : b d e ## , cette analyse est faite par (L_1)

L'élément (b) de (L_1) va reconnaître le premier élément de (b d e ##) , après le passage dans (b) il reste à analyser : d e ## , cette analyse est faite par (L_2)

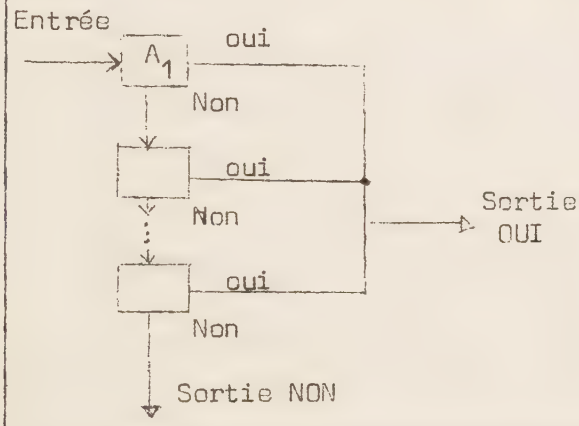
Le premier élément est reconnu par (d) de (L_2) , il reste e ## , celui-ci est reconnu par (e) de (L_2) . Alors de $\in L_2$, alors b d e $\in L_1$, alors a b d e $\in L$.

L'automate vérifie qu'il trouve bien ## en fin de mot, alors w $\in L$.

TABLEAU RECAPITULATIF DES SCHEMAS DES PARTIES D'AUTOMATES TYPES

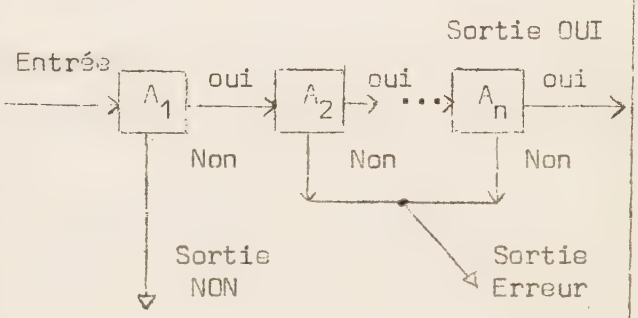
Somme :

$$S = \sum_{i=1}^{n} A_i = A_1 + A_2 + \ldots + A_n$$



Produit :

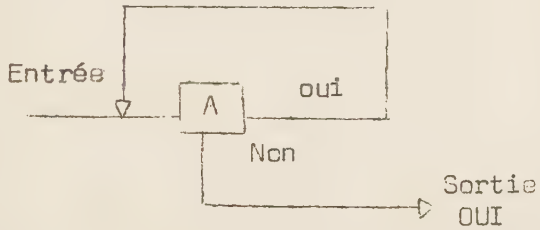
$$P = \prod_{i=1}^{n} A_i = A_1 \cdot A_2 \ldots A_n$$



Opération Etoile

$E = A^* = \mathcal{J} + A^1 + A^2 + \ldots$

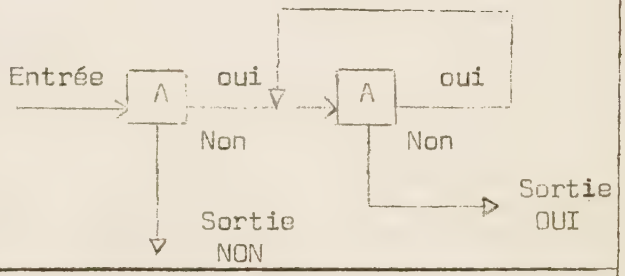
$E = A.E + I = E.A + I$



Opération itération définie

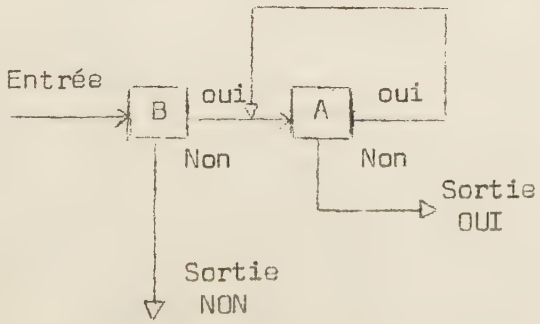
$F = A^+ = A \cdot A^*$

$F = A.F + A = F.A + A$



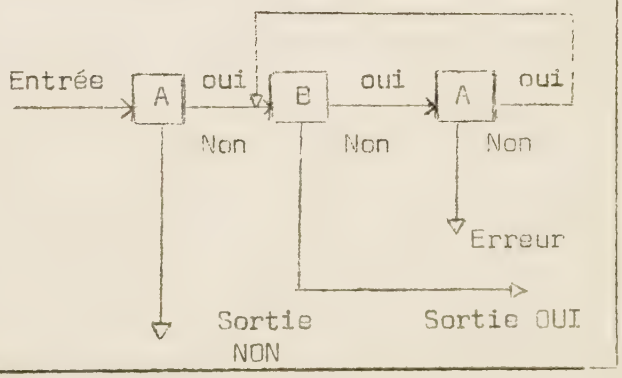
$L = L \cdot A + B$

$L = B \cdot A^*$



$L = L \cdot B \cdot A + A$

$L = A \cdot (B.A)^*$



ETUDE DES EQUATIONS SYNTAXIQUES D'ALGOL 60

L'ensemble de base ou système de générateurs libres X est l'ensemble des 116 symboles de base du rapport ALGOL 60, comprenant les lettres, les chiffres, les valeurs logiques et les délimiteurs. [cf : 6]

Le langage ALGOL est un sous-ensemble du monoïde $M = X^*$ relatif à l'ensemble de base ci-dessus, défini à partir d'un système d'équations dans l'algèbre de $\mathcal{P}(M)$.

Ce système d'équations correspond au système de formules métalinguistiques du rapport ALGOL 60 :

On remplace le signe ::= par le signe =
le signe | par le signe +
la juxtaposition par le signe .

Les symboles de base sont des constantes dans les équations. Le <u>vide</u> est identifié à $\emptyset$.

Pour éviter des confusions, tous les symboles de base seront écrits entre guillements.

Les variables libres sont obtenues, en posant :

P : Programme
B : Bloc
IC : Instruction composée
BNE : BLoc non étiqueté
ICNE : Instruction composée non étiquetée
ETB : En tête de bloc
FIC : Fin d'instruction composée
D : Déclaration
D TYP : Déclaration de Type
D TAB : Déclaration de Tableau
D PROC : Déclaration de Procédure
TYP LR : Type local ou rémanent

L TYP : Liste de type

L TAB : Liste de tableau

S TAB : Section de tableau

LPB : Liste de paires de bornes

PB : Paire de bornes

B INF : Borne inférieure

B SUP : Borne supérieure

TYP : Type

D AIG : Déclaration d'aiguillage

L AIG : Liste d'aiguillage

T PROC : Tête de procédure

PPF : Partie paramètre formel

LPF : Liste de paramètres formels

DP : Délimiteur de paramètre

P VAL : Partie valeur

P SPEC : Partie spécification

L ID : Liste d'identificateurs

C PROC : Corps de procédure

SPEC : Spécificateur

PF : Paramètre formel

I : Instruction

II : Instruction inconditionnelle

I CD : Instruction conditionnelle

I PR : Instruction pour

IB : Instruction de base

INEB : Instruction non étiquetée de base

IA : Instruction d'affectation

LPG : Liste de parties gauches

PG : Partie gauche

IAA : Instruction Aller à

I VIDE : Instruction vide

I PROC : Instruction procédure

LPE : Liste de paramètres effectifs

PPE : Partie paramètres effectifs

PE : Paramètre effectif

ISI : Instruction SI

P SI : Proposition SI

P PR : Proposition Pour

L PR : Liste de pour

EL PR : Elément d'une liste de pour

EXP : Expression

EA : Expression arithmétique

EAS : Expression arithmétique simple

TA : Terme (arithmétique)

FA : Facteur (arithmétique)

PA : Primaire arithmétique

EB : Expression booléenne

EBS : Expression booléenne simple

IMP : Implication

TB : Terme booléen

FB : Facteur booléen

SB : Secondaire booléen

PB : Primaire booléen

VAL LOG: Valeur logique

REL : Relation

OP R : Opérateur de relation

ED : Expression de désignation

EDS : Expression de désignation simple

I AIG : Indicateur d'aiguillage

O ADD : Opérateur additif

O MUL : Opérateur multiplicatif

VAR : Variable

VAR S : Variable simple

VAR I : Variable indicée

LI : Liste d'indices

EXPI : Expression en indice

ID VAR : Identificateur de variable

ID TAB : Identificateur de tableau

ID AIG : Identificateur d'aiguillage

ID PROC: Identificateur de procédure

ID ETIQ: Identificateur d'étiquette

IF : Indicateur de fonction

E : Etiquette

IDEN : Identificateur

ESS : Entier sans signe

NSS : Nombre sans signe

ND : Nombre décimal

PD : Partie décimale

FC : Facteur de cadrage

ENT : Entier

CH L : Chaîne de lettres

CH : Chaîne

CH OUV : Chaîne ouverte

CH PR : Chaîne propre

L : Lettre

C : Chiffre

EQUATIONS SYNTAXIQUES EXTRAITES DU RAPPORT ALGOL 60

On a :

(1) P = B + IC

(2) B = BNE + E . ":" . B

(3) IC = ICNE + E . ":" . IC

(4) BNE = ETB . ";" . FIC

(5) ICNE = "début" . FIC

(6) ETB = "début" . D + ETB . ";" . D

(7) FIC = IS . "fin" + IS . ";" . FIC

(8) D = D Typ + D Tab + D Aig + D Proc

(9) D Typ = Typ LR . L Typ

(10) D Tab = "Tableau" . L Tab + Typ LR . "Tableau" . L Tab

(11) D Proc = "Procédure" . T Proc . C Proc + Typ . "Procédure" . T Proc
. C Proc

(12) Typ LR = Typ + "Rémanent" . Typ

(13) L Typ = VARS + VARS . "," . L Typ

(14) Typ = "Réel" + "Entier" + "Booléen"

(15) D Aig = "Aiguillage" . Id Aig . ":=" . L Aig

(16) L Aig = ED + L Aig . "," . ED

(17) L Tab = S Tab + L Tab . "," . S Tab

(18) S Tab = Id Tab . "[" . LPB . "]" + Id Tab . "," . S Tab

(19) LPB = PB + LPB . "," . PB

(20) PB = B inf . ":" . B sup

(21) B inf = E A

(22) B sup = E A

(23) T Proc = Id Proc . PPF . ";" . P Val . P Spéc

(24) PPF = ∅ + "(" . LPF . ")"

(25) LPF = PF + LPF . DP . PF

(26) DP = "," + ")" . Ch L . ":" . "("

(27) P Val = "Valeur" . L Id . ";" + ∅

(28) P Spéc = ∅ + Spéc . L Id . ";" + P Spéc . L Id . ";"

(29) L Id = Iden + L Id . "," . Iden

(30) C Proc = IS + Code

(31) Spéc = "Chaine" + Typ + "Tableau" + Typ . "Tableau" + "Etiquette
+ "Aiguillage" + "Procédure" + Typ . "Procédure"

(32) PF = Iden

(33) IS = II + I Cd + I Pr

(34) II = IB + IC + B

(35) I Cd = ISI + ISI . "Sinon" . IS + PSI . I Pr + E . ":" . I Cd

(36) I Pr = P Pr . IS + E . ":" . I Pr

(37) IB = INEB + E . ":" . IB

(38) INEB = IA + IAA + I Vide + I Proc

(39) IA = LPG . EA + LPG . EB

(40) LPG = PG + LPG . PG

(41) PG = VAR . ":=" + Id Proc . ":="

(42) IAA = "Aller à" . ED

(43) I Proc = Id Proc . PPE

(44) PPE = ∅ + "(" . LPE . ")"

(45) LPE = PE + LPE . DP . PE

(46) PE = Ch + EXP + Id Tab + Id Aig + Id Proc

(47) EXP = EA + EB + ED

(48) I Vide = $\emptyset$

(49) ISI = PSi . II

(50) PSi = "Si" . EB . "Alors"

(51) P Pr = "Pour" . VAR . ":=" . L Pr . "Faire"

(52) L Pr = EL Pr + L Pr . "," . EL Pr

(53) EL Pr = EA + EA . "pas" . EA . "jusqu'à" . EA + EA . "tant que" . EB

(54) EA = EAS + PSi . EAS . "Sinon" . EA

(55) EAS = TA + O Add . TA + EAS . O Add . TA

(56) TA = FA + TA . O Mul . FA

(57) FA = PA + FA . "↑" . PA

(58) PA = NSS + VAR + IF + "(" . EA . ")"

(59) O Add = "+" + "-"

(60) O Mul = "X" + "/" + "÷"

(61) IF = Id Proc . PPE

(62) EB = EBS + PSi . EBS . "Sinon" . EB

(63) EBS = Imp + EBS . "≡" . Imp

(64) Imp = TB + Imp . "⊃" . TB

(65) TB = FB + TB . "V" . FB

(66) FB = SB + FB . "Λ" . SB

(67) SB = PB + "¬" . PB

(68) PB = Val log + VAR + IF + Rel + "(" . EB . ")"

(69) Val log = "Vrai" + "Faux"

(70) Rel = EAS . OR . EAS

(71) OR = "<" + "≤" + "=" + "≥" + ">" + "≠"

(72) ED = EDS + PSi . EDS . "Sinon" . ED

(73) EDS = E + I Aig + "(" . ED . ")"

(74) I Aig = Id Aig . "[" . EXPi . "]"

(75) E = Id Etiq + ESS

(76) EXPi = EA

(77) VAR = VARS + VARi

(78) VARS = Id VAR

(79) VARi = Id Tab . "[" . Li . "]"

(80) Li = EXPi + Li . "," . EXPi

(81) Id VAR = Iden

(82) Id Tab = Iden

(83) Id Aig = Iden

(84) Id Proc = Iden

(85) Id Etiq = Iden

(86) Iden = L + Iden . L + Iden . C

(87) ESS = C + ESS . C

(88) NSS = ND + FC + ND . FC

(89) ND = ESS + PD + ESS . PD

(90) PD = "." . ESS

(91) FC = "10" . Ent

(92) Ent = ESS + "+" . ESS + "-" . ESS

(93) Ch L = L + Ch L . L

(94) Ch = "(" . Ch ouv . ")"

(95) Chouv = Ch pr + "(" . Chouv . ")" + ch ouv . ch ouv

(96) Ch pr = Séquence de symboles autres que "(" ou ")" + $\emptyset$

(97) L = "A" + "B" + "C" + ... + "Y" + "Z" +
"a" + "b" + "c" + ... + "y" + "z"

(98) C = "0" + "1" + "2" + ... + "9"

CONVENTIONS DE SIMPLIFICATION

Les équations (12), (26), (75), (93), (94) et (95) sont modifiées dans un but de simplification, de la manière suivante :

(12) Typ LR = Typ (pas de rémanence)

(26) DP = "," (pas de chaînes de lettres dans les délimiteurs de paramètres)

donc l'équation (93) disparaît

(75) E = Id Etiq (pas d'étiquettes numériques)

(94) Ch = "(" . Ch pr . ")" (pas de chaines imbriquées)

donc l'équation (95) disparaît

SIMPLIFICATION DES EQUATIONS SYNTAXIQUES

Equations (1), (2) et (3)

On a : B = BNE + E . ":" . B

ou : B = (E . ":")* . BNE

En posant, LE : Liste d'étiquettes

$$\boxed{LE = (E\ .\ ":")^*} \qquad (2')$$

On a : B = LE . BNE

De même : IC = ICNE + E . ":" . IC

ou : IC = LE . ICNE

Alors P = B + IC

en remplaçant B et IC par leurs expressions s'écrit :

P = LE . BNE + LE . ICNE

En mettant LE en facteur ;

P = LE . (BNE + ICNE)

En posant, PNE : Programme non étiqueté

PNE = BNE + ICNE (a)

On a $\boxed{P = LE\ .\ PNE}$ (1')

Equations (4), (5) et (6)

On a ; ETB = "début" . D + ETB . ";" . D

ou : ETB = "début" . D .(";" . D)*

Alors, BNE = ETB . ";" . FIC

en remplaçant ETB par son expression, s'écrit :

$$BNE = \text{"début"} \,.\, D \,.\, (\text{";"} \,.\, D)^{*} \,.\, \text{";"} \,.\, FIC$$

ou $$BNE = \text{"début"} \,.\, (D \,.\, \text{";"})^{*} \,.\, D \,.\, \text{";"} \,.\, FIC$$

soit $$BNE = \text{"début"} \,.\, (D \,.\, \text{";"})^{+} \,.\, FIC$$

De plus, on a ; $ICNE = \text{"début"} \,.\, FIC$

Alors, l'équation (a), en remplaçant BNE et ICNE par leurs expressions, s'écrit :

$$PNE = \text{"début"} \,.\, (D \,.\, \text{";"})^{+} \,.\, FIC + \text{"début"} \,.\, FIC$$

en mettant "début" en facteur :

$$PNE = \text{"début"} \,.\, ((D \,.\, \text{";"})^{+} \,.\, FIC + FIC)$$

ou

$$PNE = \text{"début"} \,.\, (D \,.\, \text{";"})^{*} \,.\, FIC$$

En posant, LD ; Liste de déclarations

$$\boxed{LD = (D \,.\, \text{";"})^{*}}$$

on a ; $$\underline{PNE = \text{"début"} \,.\, LD \,.\, FIC} \qquad (b)$$

<u>Equation (7)</u>

On a ; $FIC = IS \,.\, \text{"Fin"} + IS \,.\, \text{";"} \,.\, \text{"Fin"}$

ou : $FIC = (IS \,.\, \text{";"})^{*} \,.\, IS \,.\, \text{"Fin"}$

En posant, LI : Liste d'instructions

$$\boxed{LI = (IS \,.\, \text{";"})^{*} \,.\, IS}$$

on a ; $FIC = LI \,.\, \text{"Fin"}$

L'équation (b), en remplaçant FIC par son expression, s'écrit :

$$\boxed{\text{PNE} = \text{"début"} \,.\, \text{LD} \,.\, \text{LI} \,.\, \text{"Fin"}} \qquad (3')$$

Nota

De (1'), (2') et (3'), on peut obtenir une seule équation P , remplaçant les 7 premières équations du rapport ALGOL 60.

On obtient :

$$\boxed{\text{P} = \text{LE} \,.\, \text{"début"} \,.\, \text{LD} \,.\, \text{LI} \,.\, \text{"Fin"}}$$

ou

$$\boxed{\text{P} = (\text{E} \,.\, \text{":"})^{*} \,.\, \text{"début"} \,.\, (\text{D} \,.\, \text{";"})^{*} \,.\, (\text{IS} \,.\, \text{";"})^{*} \,.\, \text{IS} \,.\, \text{"Fin"}}$$

Equations (8), (9), (10), (11), (12), (13), (15) et (16)

On a ; $\text{L Aig} = \text{ED} + \text{L Aig} \,.\, \text{","} \,.\, \text{ED}$

ou : $\text{L Aig} = \text{ED} \,.\, (\text{","} \,.\, \text{ED})^{*}$

Alors, l'équation (15), en remplaçant L Aig par son expression s'écrit :

$$\text{D Aig} = \text{"Aiguillage"} \,.\, \text{Id Aig} \,.\, \text{":="} \,.\, \text{ED} \,.\, (\text{","} \,.\, \text{ED})^{*}$$

On a aussi ; $\text{L Typ} = \text{VARS} + \text{VARS} \,.\, \text{","} \,.\, \text{L Typ}$

ou : $\text{L Typ} = (\text{VARS} \,.\, \text{","})^{*} \,.\, \text{VARS}$

avec $\text{VARS} = \text{Id VAR}$

En ne considérant pas la rémanence, soit en utilisant l'équation (12) modifiée :

$$\text{Typ LR} = \text{Typ}$$

en posant aussi :

D Tab St ; Déclaration de Tableau sans type

$$\boxed{\text{D Tab St} = \text{"Tableau"} \,.\, \text{L Tab}} \qquad (5')$$

D Proc St : Déclaration de procédure sans type

$$\boxed{\text{D Proc St} = \text{"Procédure"} \,.\, \text{T Proc} \,.\, \text{C Proc}} \qquad (c)$$

L'équation (8), en remplaçant chaque terme par son expression, et en mettant Typ en facteur, s'écrit :

$$\boxed{\begin{aligned} \text{D} = \text{Typ} \,.\, (&\text{D Tab St} + \text{D Proc St} + (\text{Id VAR} \,.\, ",")^{*} \,.\, \text{Id VAR})^{+} \\ &\text{D Tab St} + \text{D Proc St} + \\ &\text{"Aiguillage"} \,.\, \text{Id Aig} \,.\, ":=" \,.\, \text{ED} \,.\, (",\text{"} \,.\, \text{ED})^{*} \end{aligned}}$$

<u>Equations (17), (18), (19), (20), (21) et (22)</u>

L'équation (20) s'écrit, en remplaçant B inf et B sup par leurs expressions :

PB = EA . ":" . EA

on a : LPB = PB + LPB . "," . PB

ou LPB = PB . ("," . PB)*

On a aussi :

S Tab = Id Tab . "[" . LPB . "]" + Id Tab . "," . S Tab

ou S Tab = (Id Tab . ",")* . Id Tab . "[" . LPB . "]"

En posant, L Id Tab : Liste d'identificateurs de tableaux

L Id Tab = (Id Tab . ",")* . Id Tab

On obtient, pour expression de S Tab :

$$\boxed{\text{S Tab} = \text{L Id Tab} \,.\, \text{"[\,"} \,.\, \text{LPB} \,.\, \text{"]"}}$$

On a aussi :

$$\text{L Tab} = \text{S Tab} + \text{L Tab} \,.\, \text{","} \,.\, \text{S Tab}$$

ou $\quad \text{L Tab} = \text{S Tab} \,.\, (\text{","} \,.\, \text{S Tab})^*$

On obtient alors :

$$\text{D Tab St} = \text{"Tableau"} \,.\, \text{L Tab} \qquad (5')$$

avec

$$\text{L Tab} = \text{S Tab} \,.\, (\text{","} \,.\, \text{Stab})^*$$

$$\text{S Tab} = \text{L Id Tab} \,.\, \text{"[\,"} \,.\, \text{LPB} \,.\, \text{"]"}$$

$$\text{L Id Tab} = (\text{Id Tab} \,.\, \text{","})^* \,.\, \text{Id Tab}$$

$$\text{LPB} = \text{PB} \,.\, (\text{","} \,.\, \text{PB})^*$$

$$\text{PB} = \text{EA} \,.\, \text{":"} \,.\, \text{EA}$$

<u>Equations (23), (24), (25), (26), (27), (28), (29), (30), (31) et (32)</u>

On a : $\text{LPF} = \text{PF} + \text{LPF} \,.\, \text{DP} \,.\, \text{PF}$

ou : $\text{LPF} = \text{PF} \,.\, (\text{DP} \,.\, \text{PF})^*$

En utilisant l'équation (26) modifiée :

$$\text{DP} = \text{","}$$

on obtient

$$\text{LPF} = \text{PF} \,.\, (\text{","} \,.\, \text{PF})^*$$

avec $\quad \text{PF} = \text{Iden}$

$$\text{LPF} = \text{Iden} \,.\, (\text{","} \,.\, \text{Iden})^*$$

On a aussi :

L Id = Iden + L Id . "," . Iden

ou L Id = Iden . ("," . Iden)*

d'où <u>LPF = L Id</u>

On a :

P Spéc = $\emptyset$ + Spéc . L Id . ";" + P Spéc . Spéc . L Id . ";"

ou P Spéc = ($\emptyset$ + Spéc . L Id . ";") . (Spéc . L Id . ";")*

ou encore :

P Spéc = (Spéc . L Id . ";")*

comme :

T Proc = Id Proc . PPF . ";" . P Val . P Spéc

et PPF = $\emptyset$ + "(" . LPF . ")"

On obtient alors pour (c) :

(6') D Proc St = "Procédure" . Id Proc . ($\emptyset$ + "(" . L Id . ")") . ";" .
("Valeur" . L Id . ";" + $\emptyset$) .
(Spéc . L Id . ";")* . (IS + Code)

avec L Id = Iden . (";" . Iden)*
Spéc = "chaîne" + "Etiquette" + "Aiguillage" + Typ +
($\emptyset$ + "Tableau" + "Procédure") + "Tableau" + "Procédure"

Equations (33), (34), (35), (36) et (37)

On a : IB = INEB + E . ":" . IB

ou : IB = LE . INEB

L'équation (34) : II = IB + IC + B

en remplaçant IC + B par P défini par (1') et en mettant LE en facteur, s'écrit :

II = LE . (PNE + INEB)

En posant, IINE : Instruction inconditionnelle non étiquetée

IINE = PNE + INEB (d)

on obtient

II = L E . IINE (e)

On a :

I Cd = ISi + ISi . "Sinon" . IS + PSi . I Pr + E . ":" . I Cd

ou

I Cd = LE . (ISi + ISi . "Sinon" . IS + PSi . I Pr)

En posant, I Cd NE : Instruction conditionnelle non étiquetée

I Cd NE = ISi + ISi . "Sinon" . IS + PSi . I Pr (h)

on obtient : I Cd = LE . I Cd NE (f)

On a aussi :

I Pr = PPr . IS + E . ":" . IPr

ou I Pr = LE . PPr . IS

En posant, I Pr NE : Instruction pour non étiquetée

I Pr NE = PPr . IS (i)

on a :

I Pr = L E . I Pr NE (g)

Enfin l'équation (33), en utilisant (e), (f) et (g), s'écrit :

IS = LE . (IINE + I Cd NE + I Pr NE)

soit en posant, INE : Instruction non étiquetée

$$\boxed{\text{INE} = \text{IINE} + \text{I Cd NE} + \text{I Pr NE}} \qquad (8')$$

on a : $\boxed{\text{IS} = \text{LE . INE}}$ (7')

<u>Equations (38), (39), (40), (41), (42), (43), (44), (45), (46), (47) et (48</u>

On a : IA = LPG . EA + LPG . EB

soit en mettant LPG en facteur

IA = LPG . (EA + EB)

On a aussi :

LPG = PG + LPG . PG

ou LPG = PG . PG^{*} = PG^{+}

comme PG = VAR . ":=" + Id Proc . ":="

ou PG = (VAR + Id Proc) . ":="

on obtient

LPG = $((\text{VAR} + \text{Id Proc}) \text{ . ":="})^{+}$

On a : LPE = PE + LPE . DP . PE

En utilisant l'équation (26) modifiée :

DP = ","

on a : LPE = PE . $(\text{"," . PE})^{*}$

Comme I Proc = Id Proc . PPE

et PPE = $\mathcal{J}$ + "(". LPE . ")"

On obtient :

$$I\ Proc = Id\ Proc\ .\ (\mathfrak{J} + "(" \ .\ PE\ .\ (","\ .\ PE)^{*}\ .\ ")")$$

En remplaçant IA, IAA, I Vide, I Proc dans l'équation (38), puis INEB dans l'équation (d), on obtient :

$$\boxed{\begin{aligned} IINE = PNE + LPG\ .\ (EA + EB) + "Aller\ à"\ .\ ED + \mathfrak{J} + \\ Id\ Proc\ .\ (\mathfrak{J} + "(" \ .\ PE\ .\ (","\ .\ PE)^{*}\ .\ ")") \end{aligned}} \qquad (9')$$

avec

$$\boxed{LPG = ((VAR + Id\ Proc)\ .\ ":=")^{+}}$$

$$\boxed{PE = Ch + EA + EB + ED + Id\ Tab + Id\ Aig + Id\ Proc}$$

<u>Equations (49) et (50)</u>

En utilisant (49), l'équation (h) s'écrit :

$$I\ Cd\ NE = PSi\ .\ II + PSi\ .\ II\ .\ "Sinon"\ .\ IS + PSi\ .\ I\ Pr$$

En mettant PSi en facteur, on a ;

$$I\ Cd\ NE = PSi\ .\ (II + II\ .\ "Sinon"\ .\ IS + I\ Pr)$$

En mettant II en facteur et en utilisant (50), (e), (g) et (7'), on obtient :

$$\boxed{\begin{aligned} I\ Cd\ NE = "Si"\ .\ EB\ .\ "Alors"\ .\ LE\ .\ (IINE\ .\ (\mathfrak{J} + "Sinon"\ .\ LE\ .\ INE) \\ + I\ Pr\ NE) \end{aligned}} \qquad (10')$$

<u>Equations (51), (52) et (53)</u>

On a : $L\ Pr = EL\ Pr + L\ Pr\ .\ ","\ .\ EL\ Pr$

ou : $L\ Pr = EL\ Pr\ .\ (","\ .\ EL\ Pr)^{*}$ (j)

L'équation (53), en mettant EA en facteur, s'écrit :

$$\boxed{\text{EL Pr} = \text{EA} \,.\, (\emptyset + \text{"pas"} \,.\, \text{EA} \,.\, \text{"jusqu'à"} \,.\, \text{EA} + \text{"tant que"} \,.\, \text{EB})} \quad (k)$$

En utilisant (51), (7') et (j), l'équation (i) s'écrit :

$$\boxed{\begin{array}{l}\text{I Pr NE} = \text{"Pour"} \,.\, \text{VAR} \,.\, \text{":="} \,.\, \text{EL Pr} \,.\, (\text{","} \,.\, \text{EL Pr})^{*} \\ \qquad\qquad .\, \text{"Faire"} \,.\, \text{LE} \,.\, \text{INE}\end{array}} \quad (11')$$

avec EL Pr donné par l'équation (k)

<u>Equations (55), (56), (57), (58), (59), (60) et (61)</u>

On a : EAS = TA + O Add . TA + EAS . O Add . TA

ou $$\boxed{\text{EAS} = (\text{TA} + \text{O Add} \,.\, \text{TA}) \,.\, (\text{O Add} \,.\, \text{TA})^{*}} \quad (13')$$

avec O Add = "+" + "−"

L'équation (56) se réduit à

$$\boxed{\text{TA} = \text{FA} \,.\, (\text{O Mul} \,.\, \text{FA})^{*}} \quad (14')$$

avec O Mul = "x" + "/" + "÷"

L'équation (57) se réduit à

$$\boxed{\text{FA} = \text{PA} \,.\, (\text{"↑"} \,.\, \text{PA})^{*}} \quad (15')$$

En remplaçant PPE dans (61) et IF dans (58), on obtient :

$$\boxed{\begin{array}{l}\text{PA} = \text{NSS} + \text{VAR} + \text{Id Proc} \,.\, (\emptyset + \text{"("} \,.\, \text{PE} \,.\, (\text{","} \,.\, \text{PE})^{*} \,.\, \text{")"}) \\ \qquad + \text{"("} \,.\, \text{EA} \,.\, \text{")"}\end{array}} \quad (16')$$

Equations (63), (64), (65), (66), (68), (69),(70), et (71)

Les équations (63), (64), (65) et (66) s'écrivent :

EBS = Imp . ("≡" . Imp)*	(18')
Imp = TB . ("⊃" . TB)*	(19')
TB = FB . ("V" . FB)*	(20')
FB = SB . ("∧" . SB)*	(21')

On a Val Log = "Vrai" + "Faux"

Rel = EAS . OR . EAS

d'où l'équation (68), en remplaçant Val Log, Rel et IF par leurs expressions, s'écrit :

PB = "Vrai" + "Faux" + VAR + Id Proc . (∂ + "(" . PE . (",," . PE)* . ")") + EAS . OR . EAS + "(" . EB . ")"

avec OR donné par (71)

Equations (73), (74), (75) et (76)

On a . EDS = E + I Aig + "(" . EB . ")"

En utilisant l'équation (75) modifiée :

E = Id Etiq

et avec :

I Aig = Id Aig . "[" . EX Pi . "]"

où EXPi = EA

on obtient :

$$\boxed{EDS = Id\ Etiq + Id\ Aig\ .\ "[" \ .\ EA\ .\ "]" + "(" \ .\ ED\ .\ ")"} \qquad (25')$$

Equations (79)et (80)

On a : $VARi = Id\ Tab\ .\ "[" \ .\ Li\ .\ "]"$

$Li = EXPi + Li\ .\ "," \ .\ EXPi$

ou $Li = EXPi\ .\ ("," \ .\ EXPi)^*$

Comme $EXPi = EA$, on obtient :

$$\boxed{VAR\ i = Id\ Tab\ .\ "[" \ .\ EA\ .\ ("," \ .\ EA)^* \ .\ "]"}$$

Equation (86)

On a : $Iden = L + Iden\ .\ L + Iden\ .\ C$

soit : $Iden = L + Iden\ .\ (L + C)$

ou $$\boxed{Iden = L\ .\ (L + C)^*} \qquad (26')$$

Equations (87), (88), (89), (90) et (91)

On a : $ESS = C + ESS\ .\ C$

ou : $ESS = C\ .\ C^* = C^+$

On a aussi : $ND = ESS + PD + ESS\ .\ PD$

ou : $ND = ESS\ .\ (\emptyset + PD) + PD$

et : $ENT = ESS + "+" \ .\ ESS + "-" \ .\ ESS$

ou : $ENT = ESS + ("+" + "-")\ .\ ESS$

Alors ; NSS = ND + FC + ND . FC

ou : NSS = ND . (1 + FC) + FC

s'écrit, en utilisant les équations (90) et (91) :

$$NSS = (C^{+} . (1 + "." . C^{+}) + "." C^{+}) .$$
$$(1 + "10" . (C^{+} + ("+" + "-") . C^{+})) +$$
$$"10" . (C^{+} + ("+" + "-") . C^{+})$$

EQUATIONS SYNTAXIQUES SIMPLIFIEES

(1') P = LE . PNE

(2') LE = (E . ":")*

(3') PNE = "début" . LD . LI . "Fin"

avec : LD = (D . ";")*

LI = (IS . ";")* . IS

D = Typ . D Tab St + D Proc St + (Id VAR . ",")* .

(Id VAR) + D Tab St + D Proc St +

"Aiguillage" . Id Aig . ":=" . ED . (",". ED)*

(4') Typ = "Réel" + "Entier" + "Booléen"

(5') D Tab St = "Tableau" . L Tab

avec : L Tab = S Tab . (",". S Tab)*

S Tab = L Id Tab . "[" . LPB . "]"

L Id Tab = Id Tab . (",". Id Tab)*

LPB = PB . (",". PB)*

PB = EA . ":" . EA

(6') D Proc St = "Procédure" . Id Proc . (∅ + "(" . LId . ")") .

";" . ("Valeur" . L Id . ";" + ∅) .

(Spéc . L Id . ";")* . (IS + Code)

avec ; L Id = Iden . (",". Iden)*

Spéc = "chaîne" + "Etiquette" + "Aiguillage" + Typ

(∅ + "Tableau" + "Procédure") + "Tableau" +
"Procédure"

(7') IS = LE . INE

(8') INE = IINE + I Cd NE + I Pr NE

(9') IINE = PNE + LPG . (EA + EB) + "Aller à" . ED +

∅ + Id Proc . (∅ + "(" . PE . (",") . PE)* . ")")

avec :

LPG = ((VAR + Id Proc) . ":=")+

PE = Ch + EA + EB + ED + Id Tab + Id Aig + Id Proc

Ch = "{" . Ch pr . "}"

Ch pr = séquence de symboles autres que "{" ou "}" + ∅

(10') I Cd NE = "Si" . EB . "Alors" . LE . (IINE . (∅ + "Sinon" .

LE . INE) + I Pr NE)

(11') I Pr NE = "Pour" . VAR . ":=" . EL Pr . (",") . EL Pr)* .

"Faire" . LE . INE

avec :

EL Pr = EA . (∅ + "pas" . EA . "jusqu'à" . EA +

"tant que" . EB)

(12') EA = EAS + "Si" . EB . "Alors" . EAS . "Sinon" . EA

(13') EAS = (TA + O Add TA) . (O Add . TA)*

avec : O Add = "+" + "-"

(14') TA = FA . (O Mul . F A)*

avec : O Mul = "x" + "/" + "÷"

(15') FA = PA . ("↑" . PA)*

(16') PA = NSS + VAR + Id Proc . (∅ + "(" . PE . (",") . PE)* .

")") + "(" . EA . ")"

(17') EB = EBS + "Si" . EB . "Alors" . EBS . "Sinon" . EB

(18') EBS = Imp . ("≡" . Imp)*

(19') Imp = TB . ("⇒" . TB)*

(20') TB = FB . ("V" . FB)*

(21') FB = SB . ("Λ" . SB)*

(22') SB = PB + "¬" . PB

(23') PB = "Vrai" + "Faux" + VAR + Id Proc . (∅ + "(" . PE .

("," . PE)* . ")") + EAS . OR . EAS + "(" . EB . ")"

avec :

OR = "<" + "≤" + "=" + "≥" + ">" + "≠"

VAR = Id VAR + VARi

VARi = Id Tab . "[" . EA . ("," . EA)* . "]"

(24') ED = EDS + "Si" . EB . "Alors" . EDS . "Sinon" . ED

(25') EDS = Id Etiq + Id Aig . "[" . EA . "]" + "(" . EB . ")"

(26') Iden = L . (L + C)*

(27') NSS = (C$^+$. (∅ + "." . C$^+$) + "." . C$^+$) .

(∅ + "10" . (C$^+$ + ("+" + "-") . C$^+$)) +

"10" . (C$^+$ + ("+" + "-") . C$^+$)

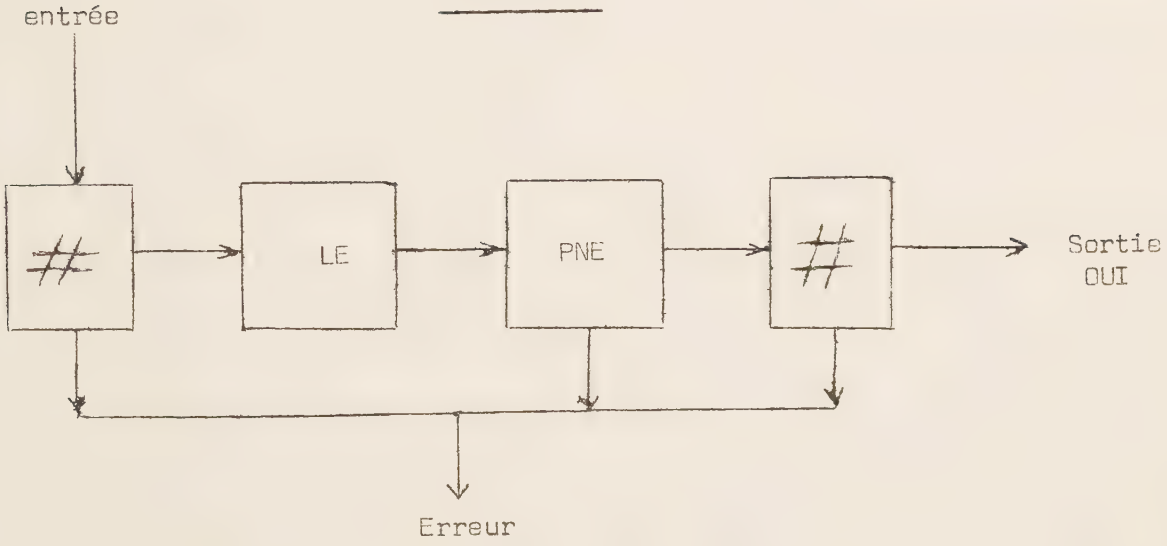
(28') L = "A" + "B" + ... + "Z" + "a" + "b" + ... + "z"

(29') C = "0" + "1" + "2" + ... + "9"

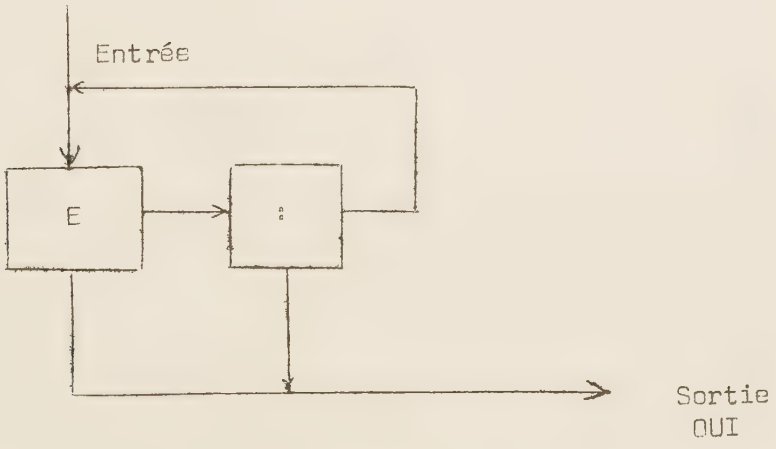
SCHEMAS

DE L'AUTOMATE DE RECONNAISSANCE ASSOCIE AU SYSTEME D'EQUATIONS SYNTAXIQUES SIMPLIFIE D'ALGOL 60

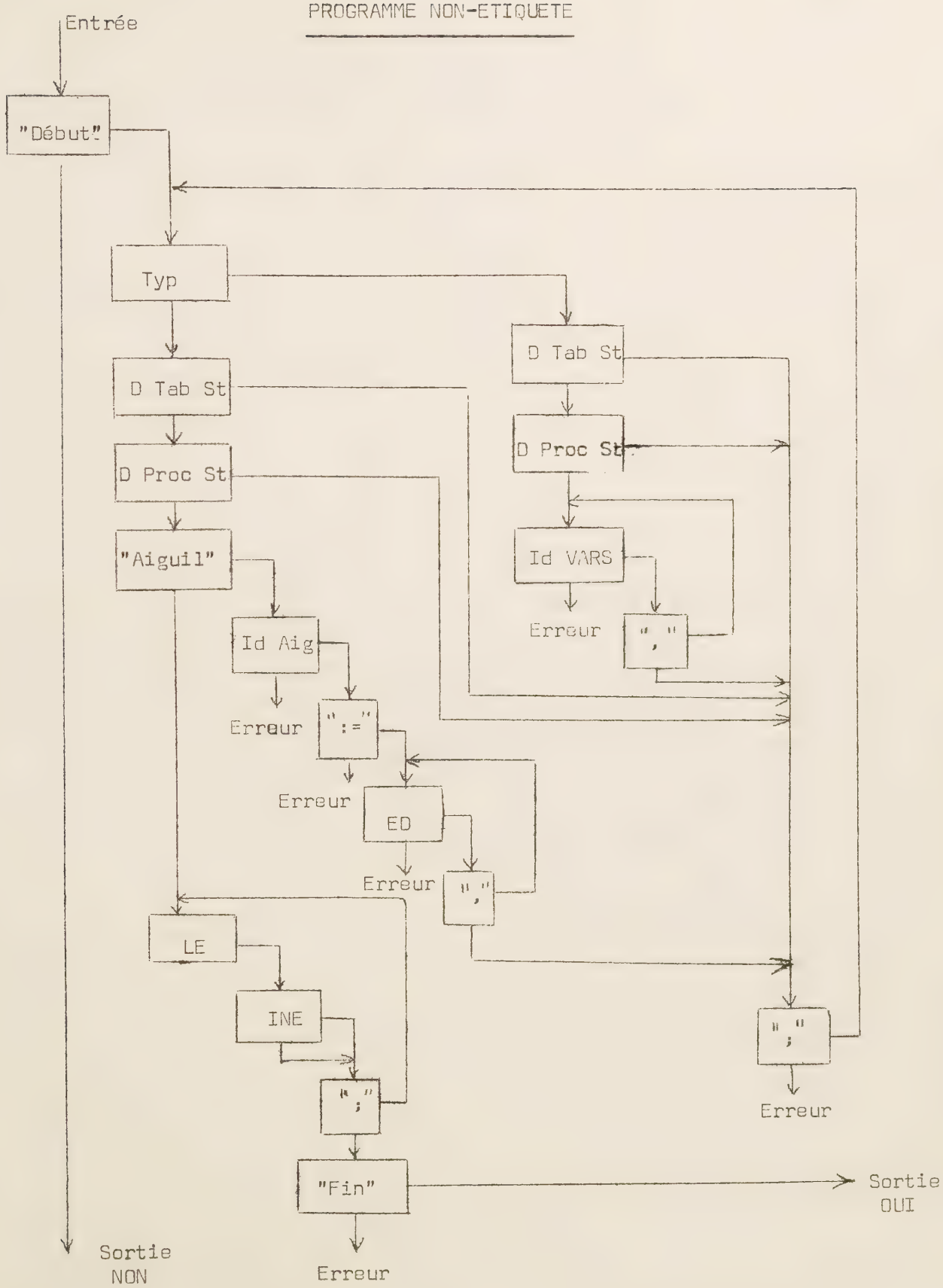
PROGRAMME



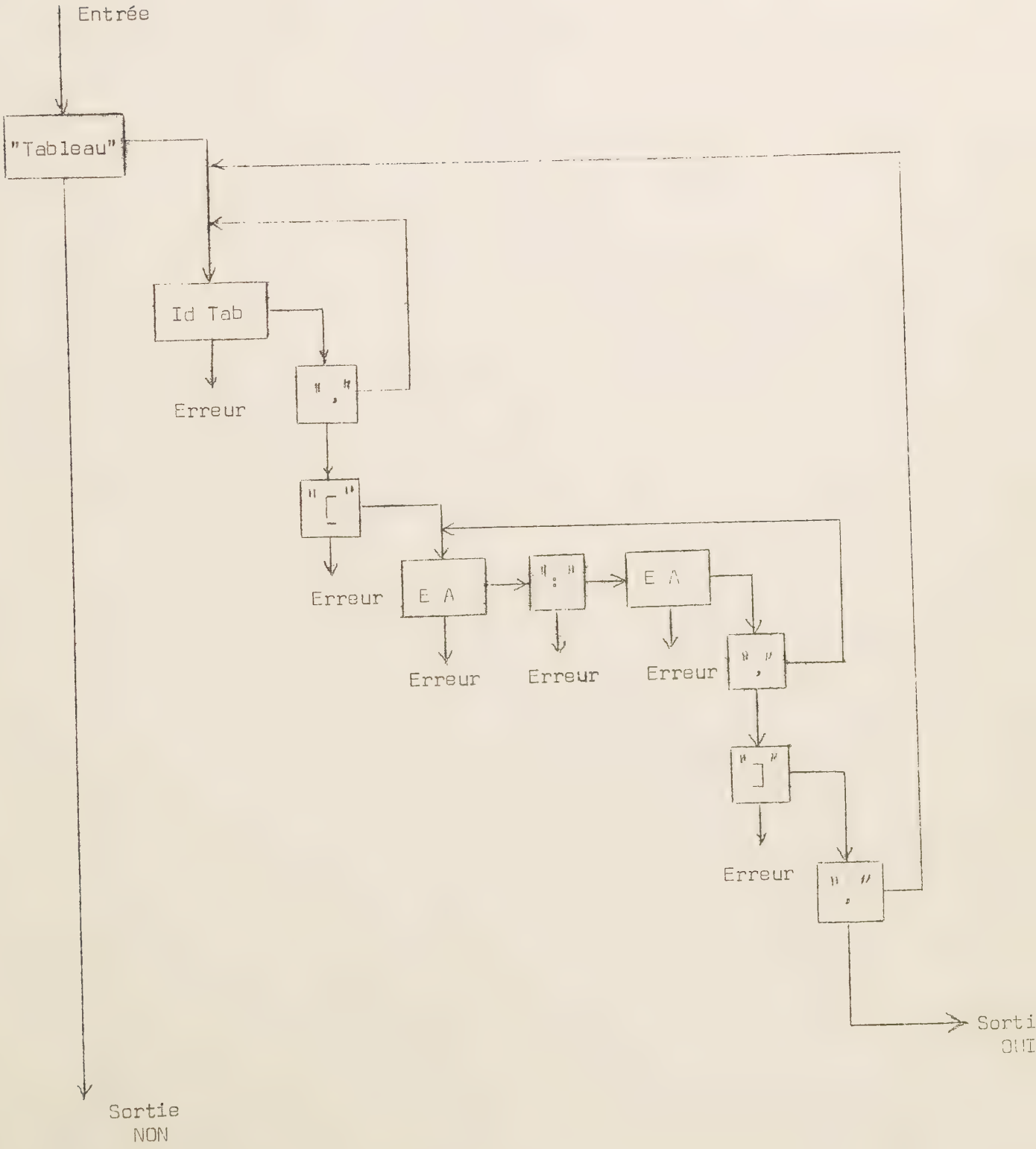
Liste d'étiquettes



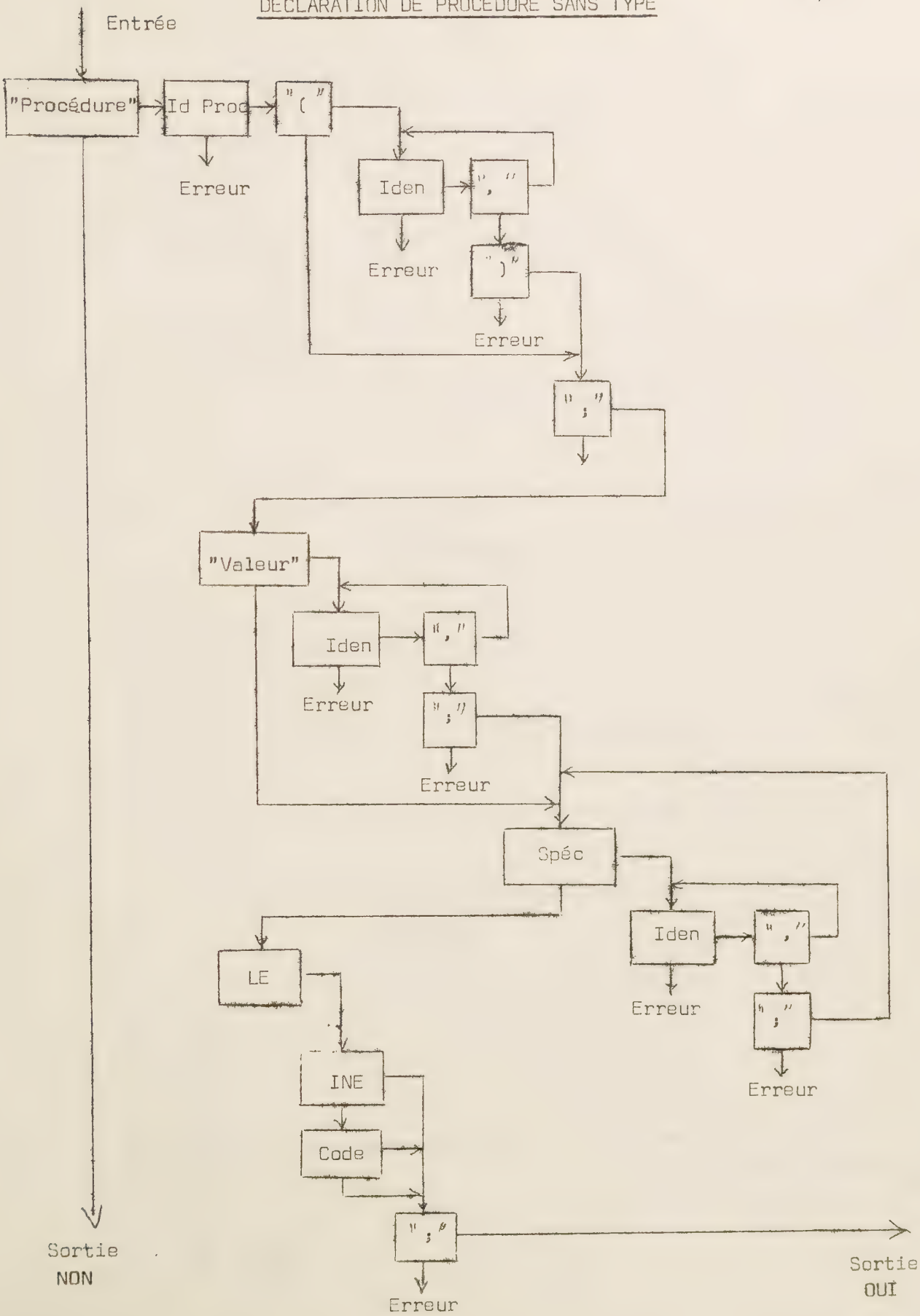
PROGRAMME NON-ETIQUETE



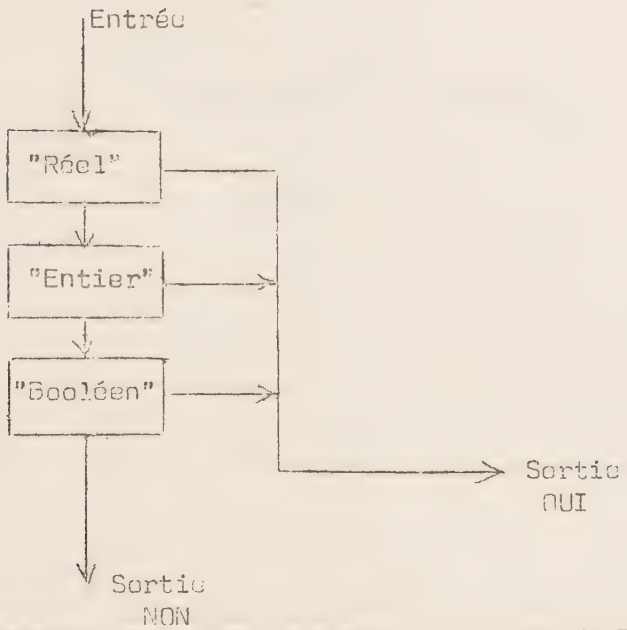
DECLARATION DE TABLEAU SANS TYPE



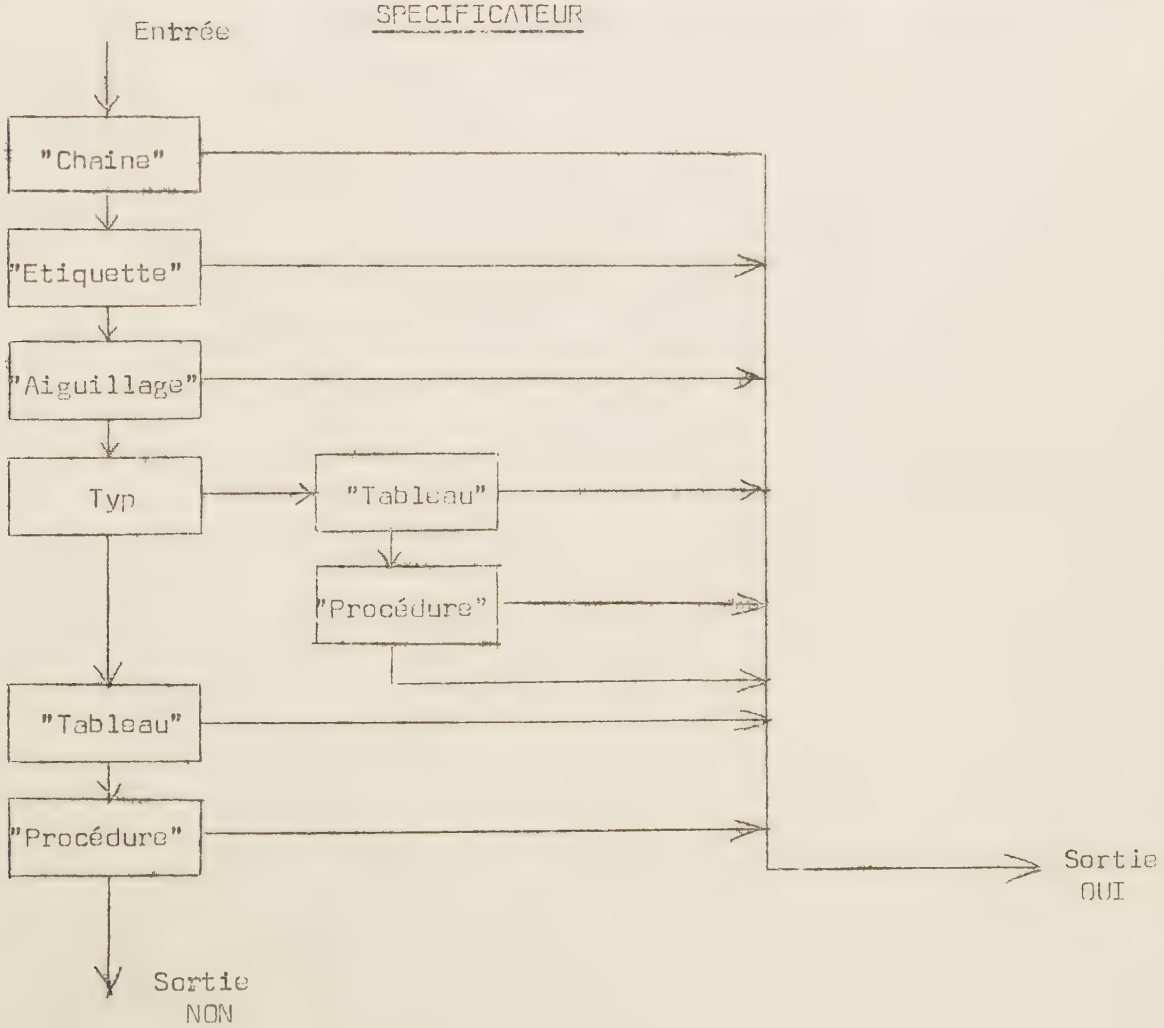
DECLARATION DE PROCEDURE SANS TYPE



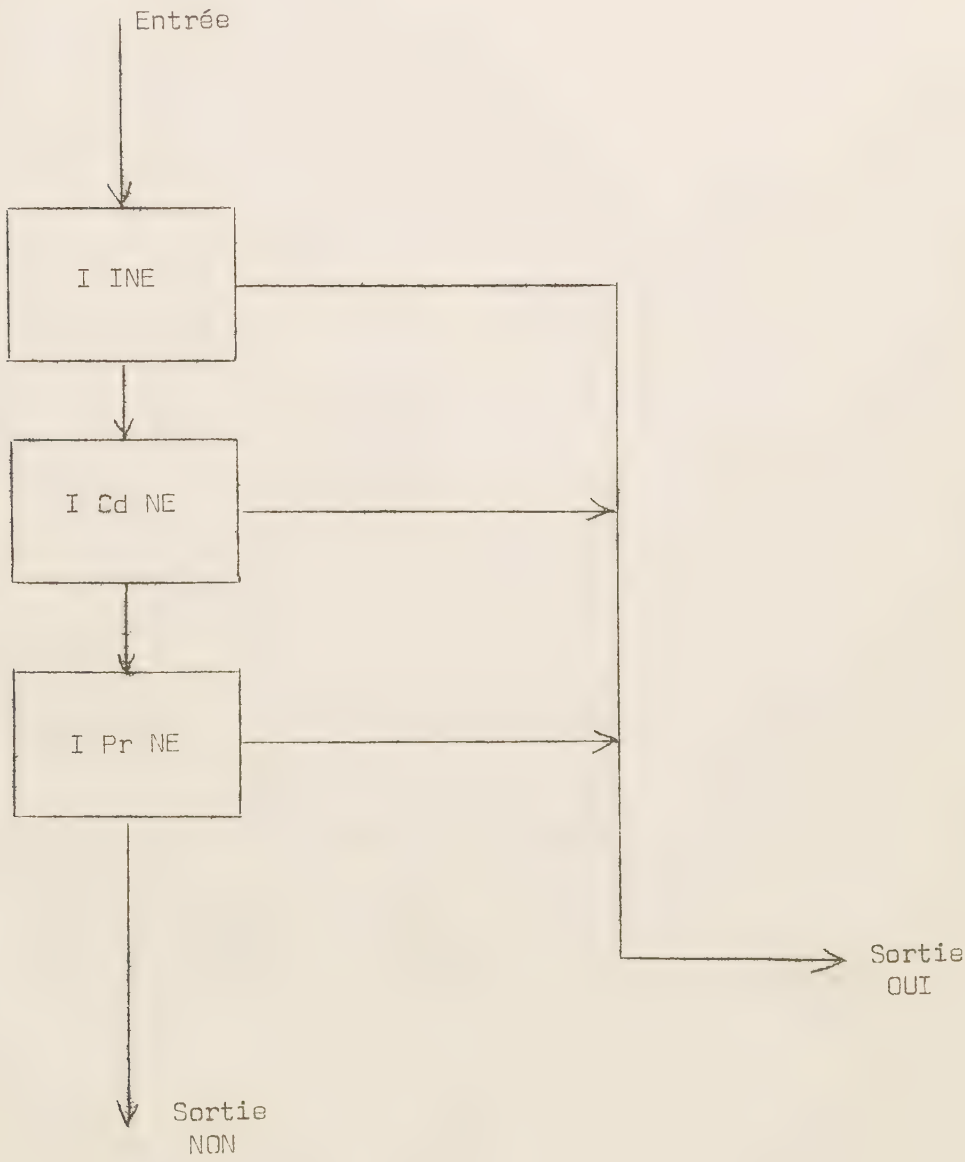
TYPE



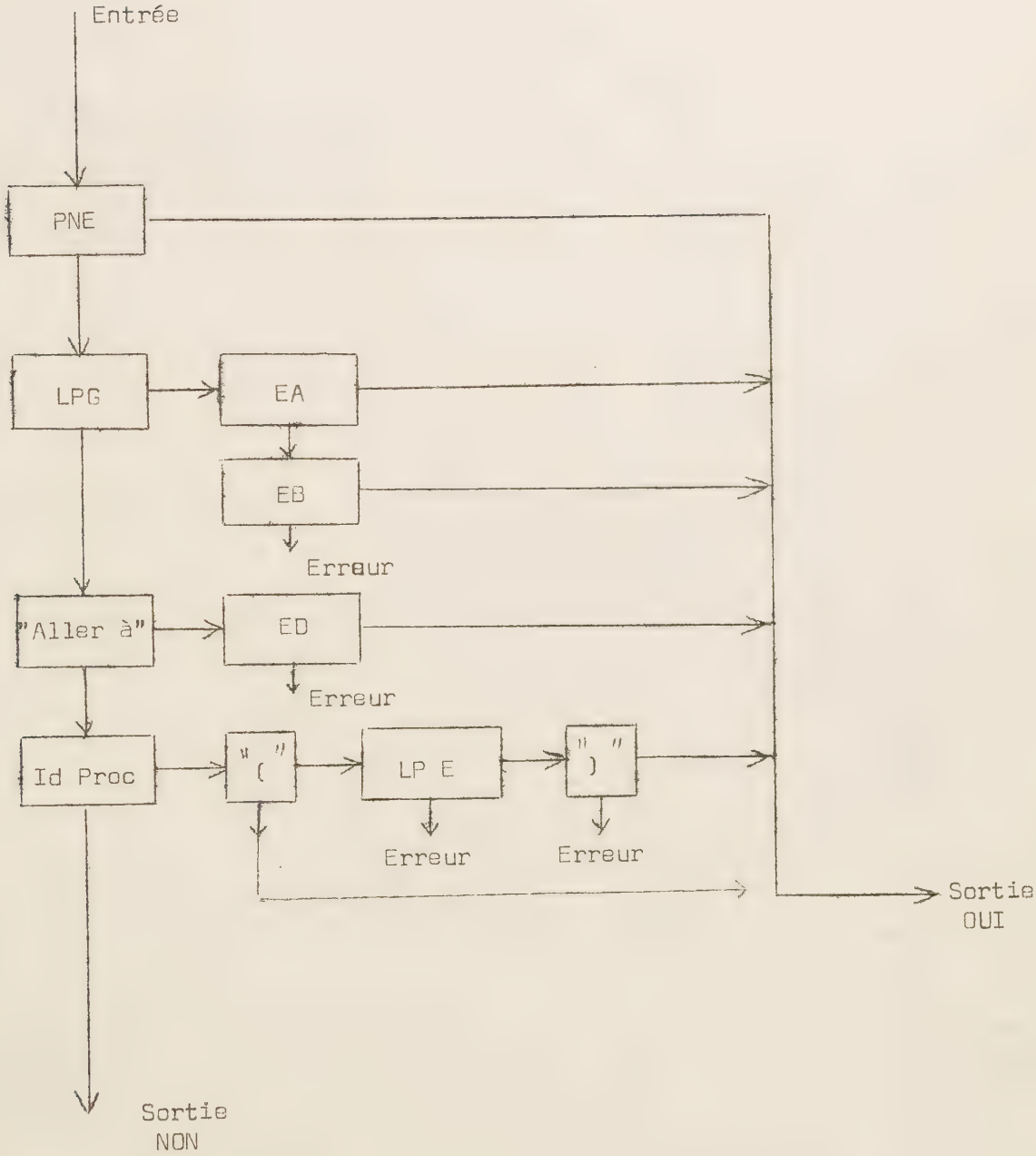
SPECIFICATEUR



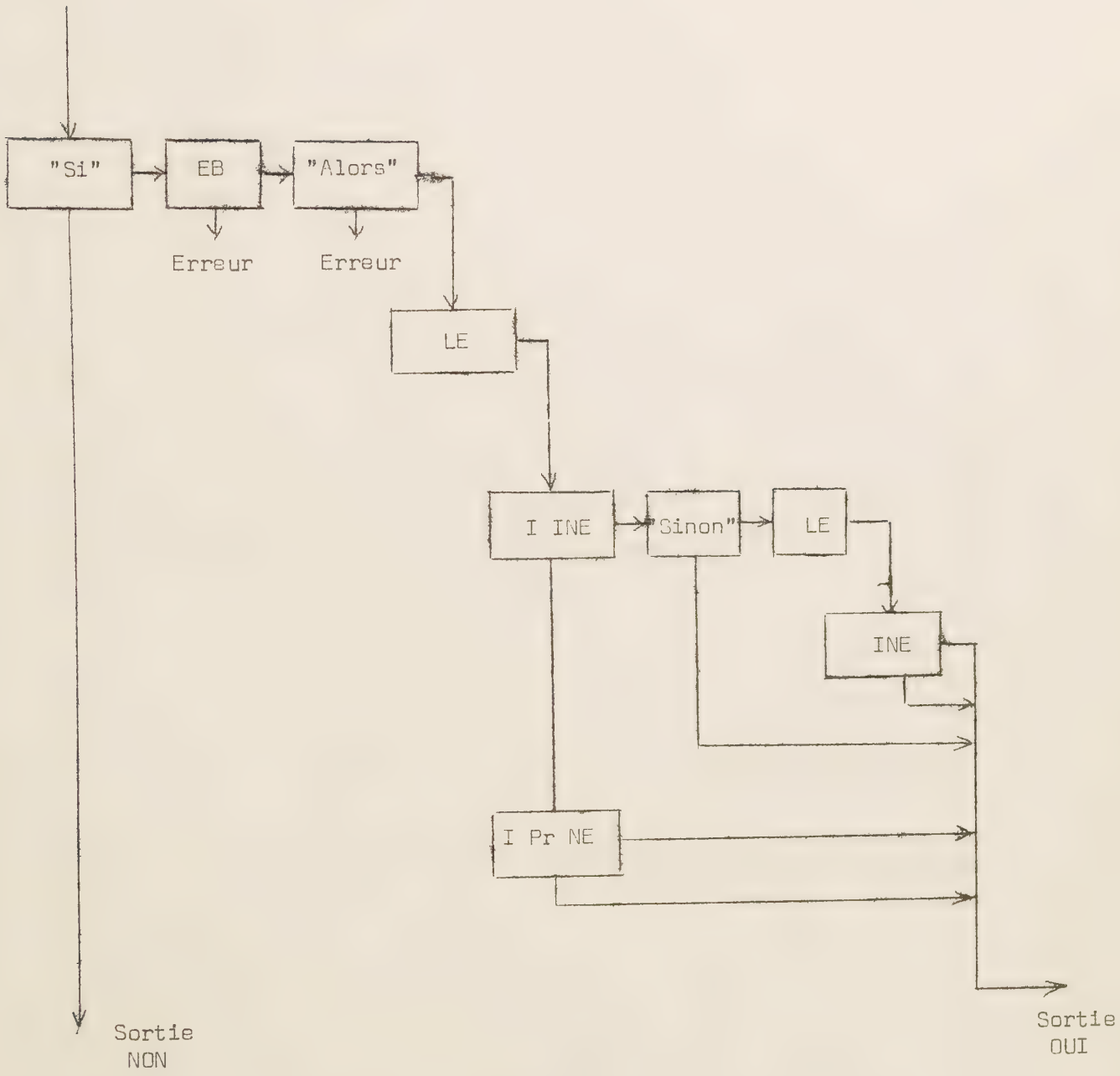
INSTRUCTION NON ETIQUETEE



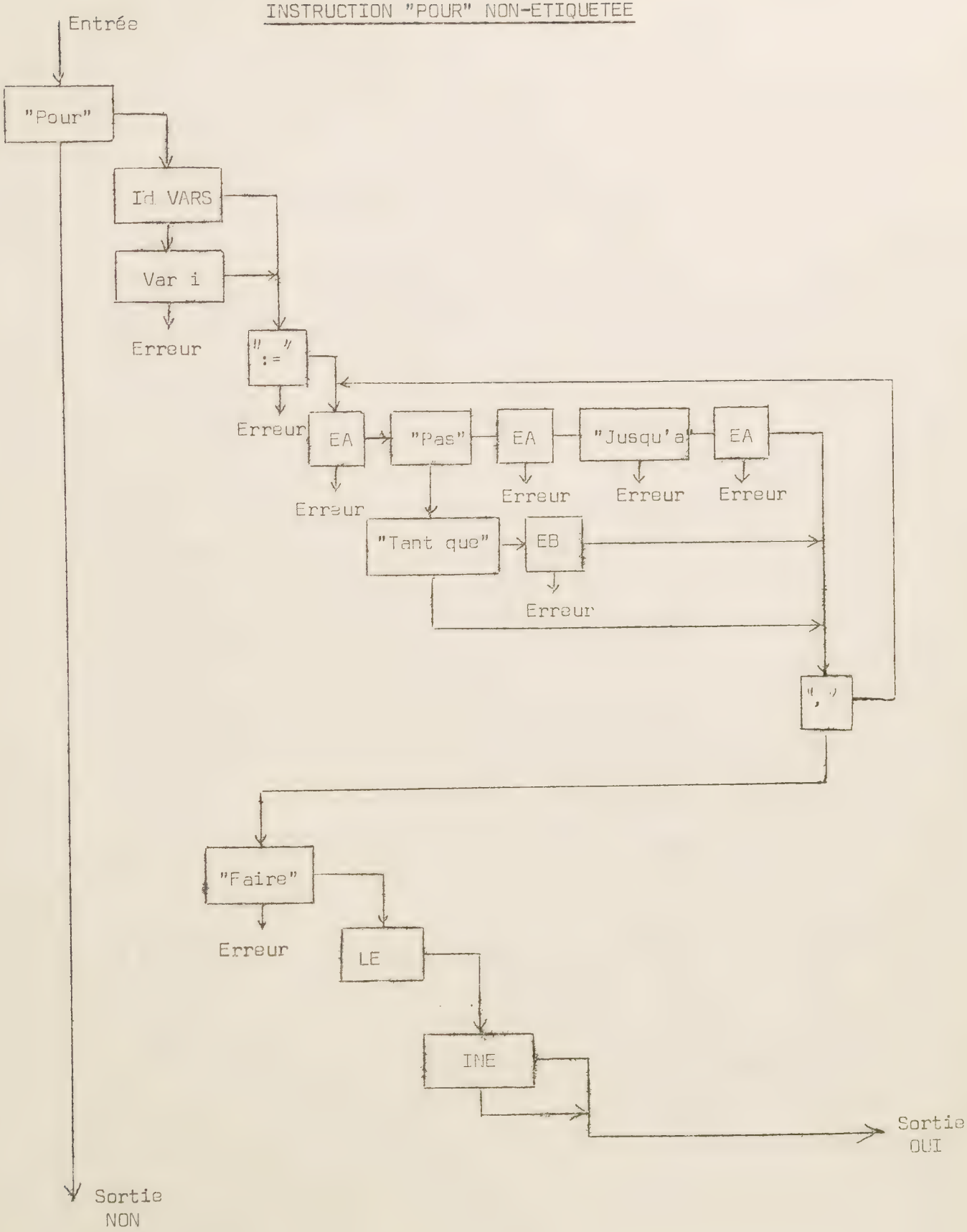
INSTRUCTION INCONDITIONNELLE NON-ETIQUETEE



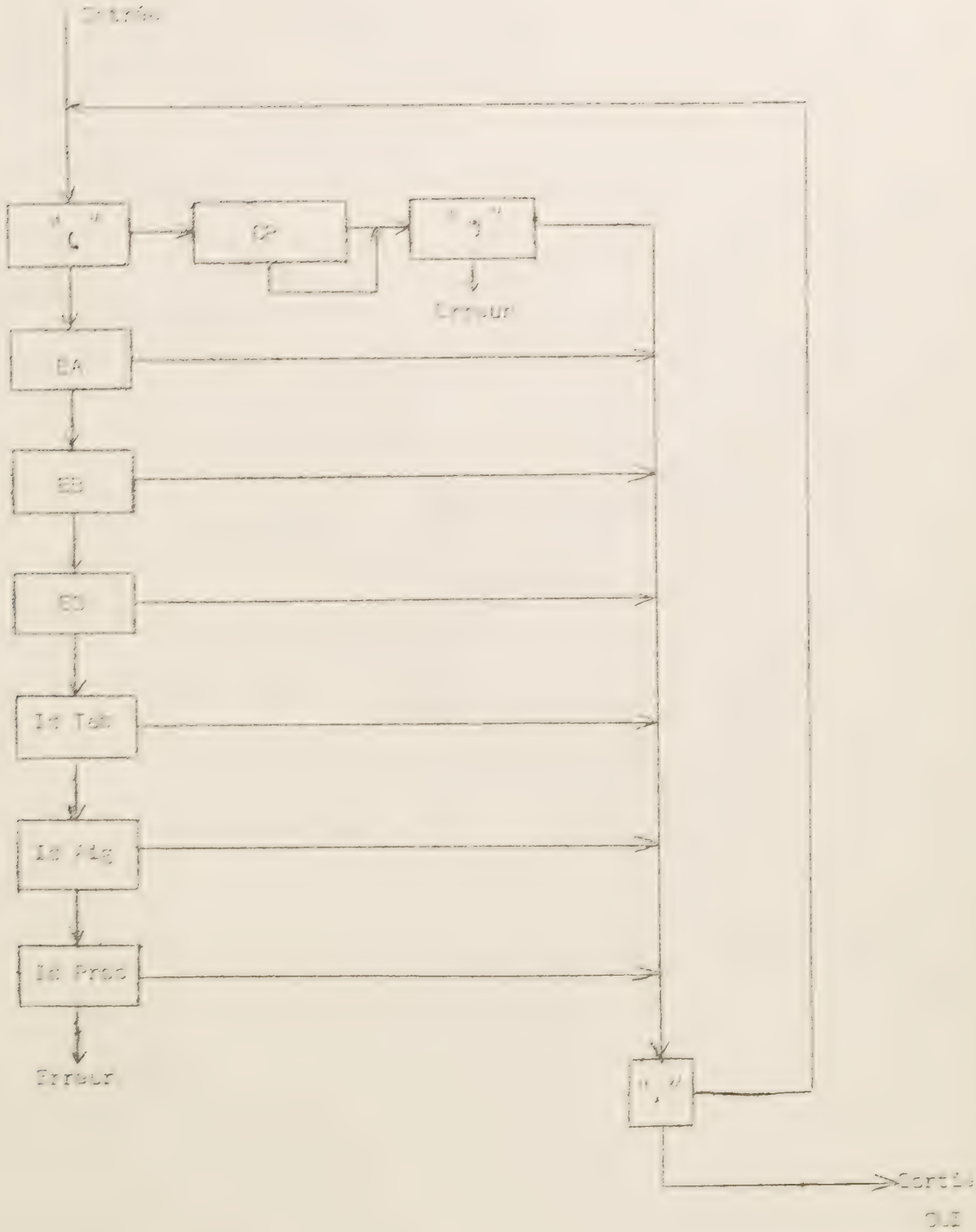
INSTRUCTION CONDITIONNELLE NON-ETIQUETEE



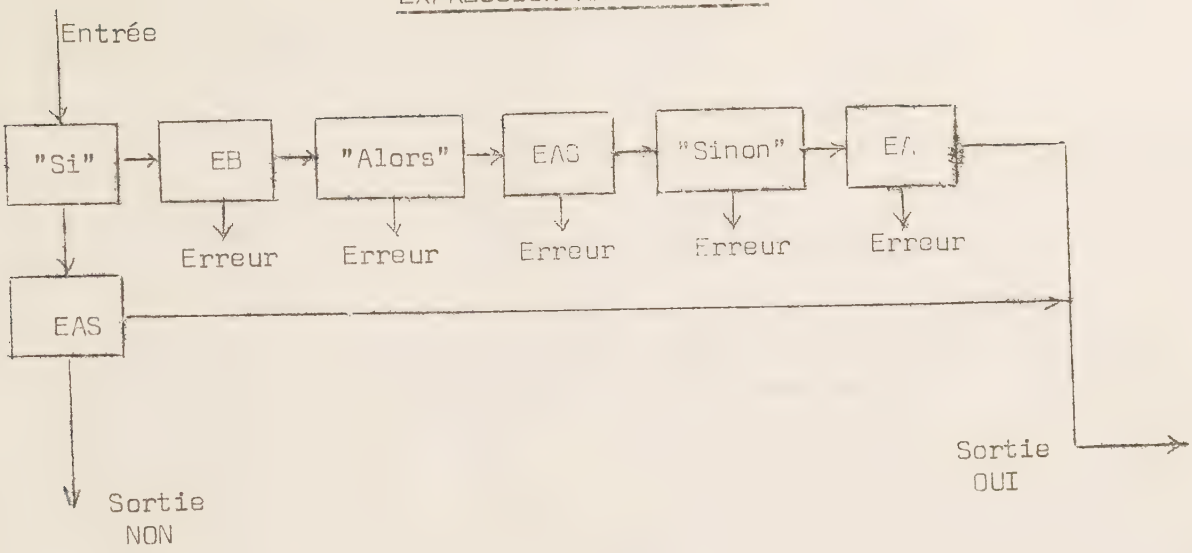
INSTRUCTION "POUR" NON-ETIQUETEE



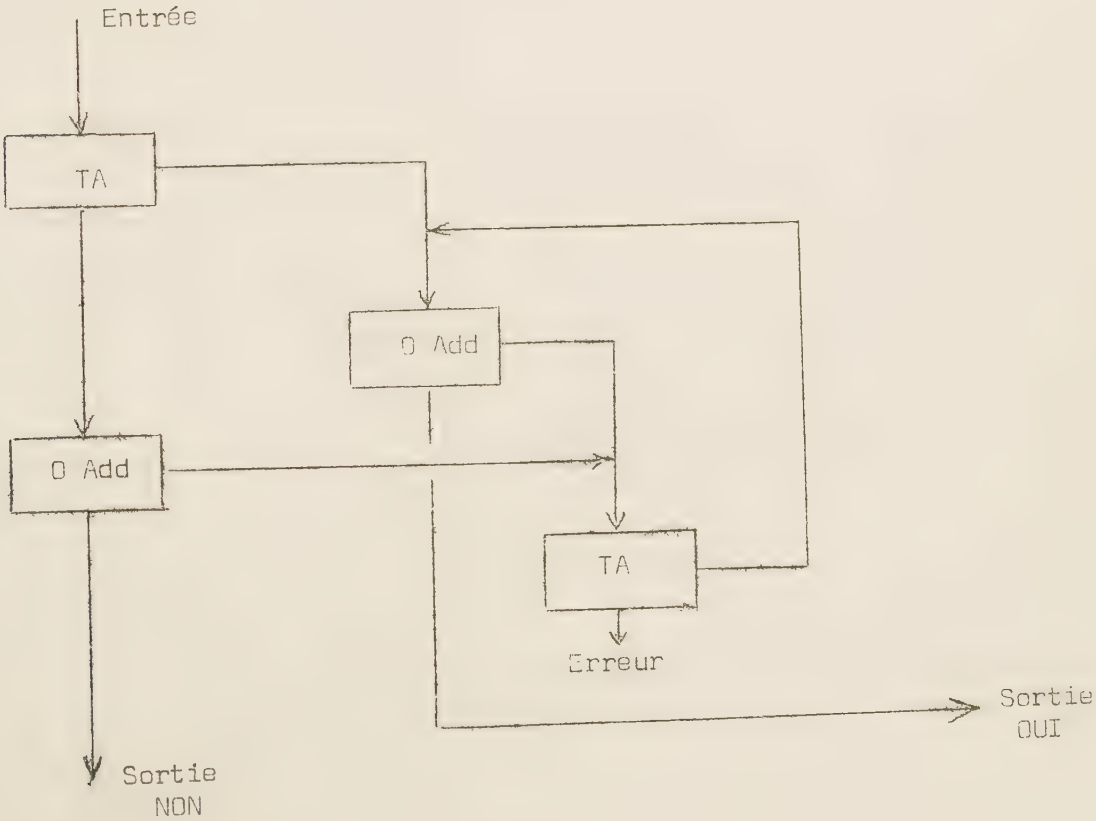
LISTE DE PARAMETRES EFFECTIFS



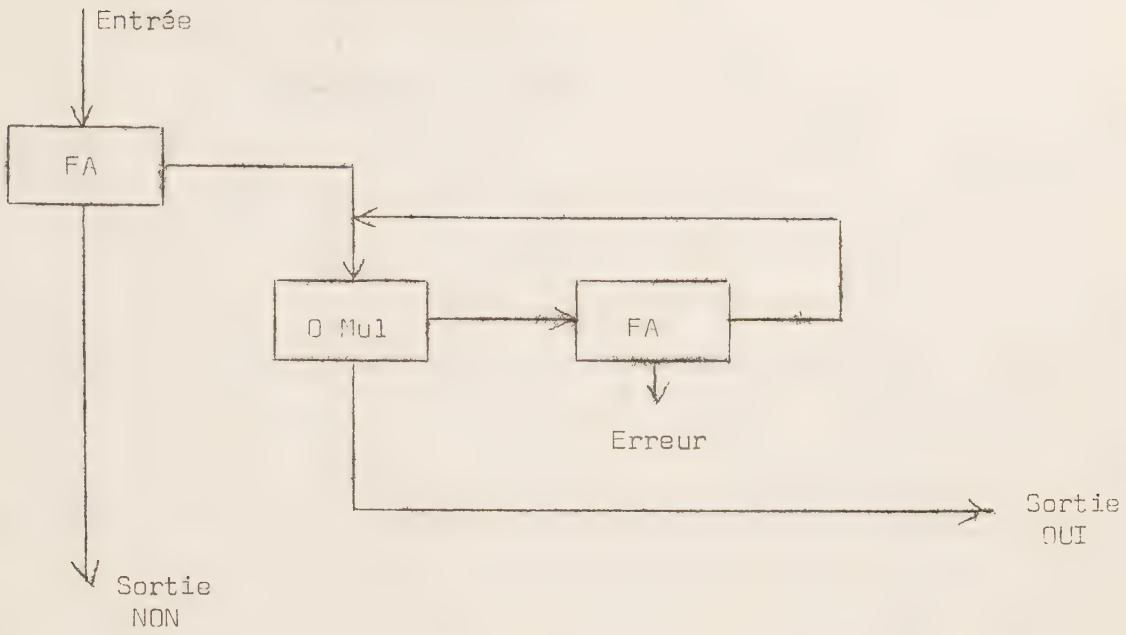
EXPRESSION ARITHMETIQUE



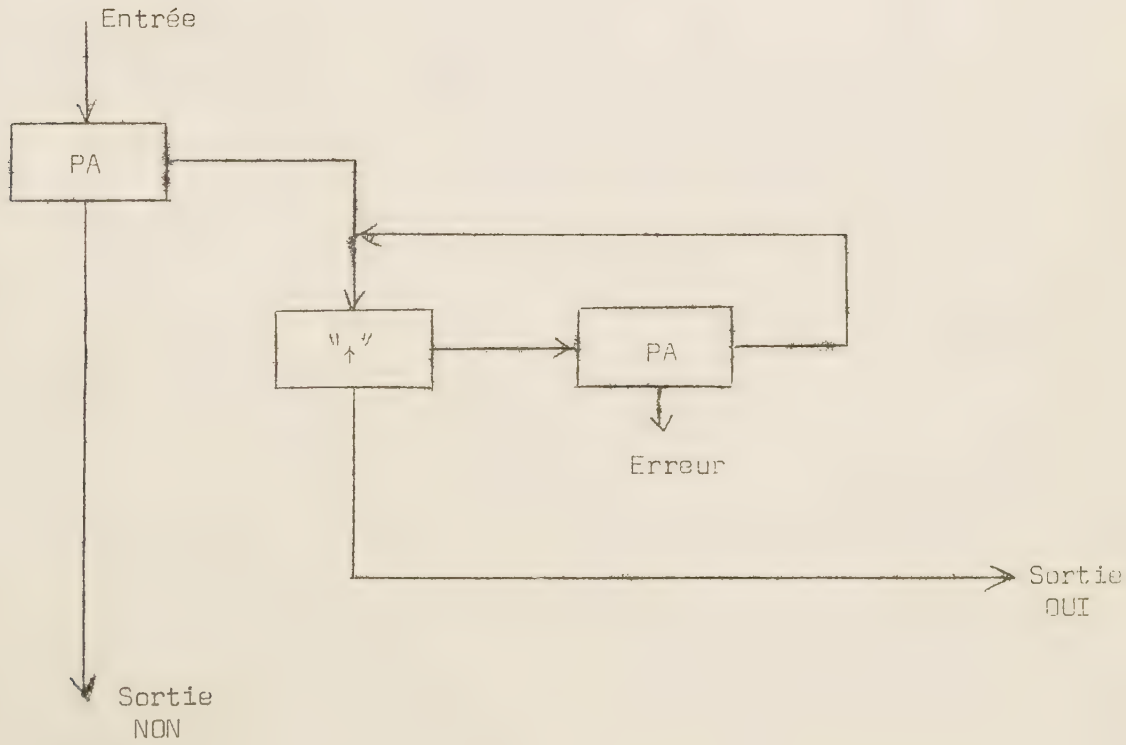
EXPRESSION ARITHMETIQUE SIMPLE



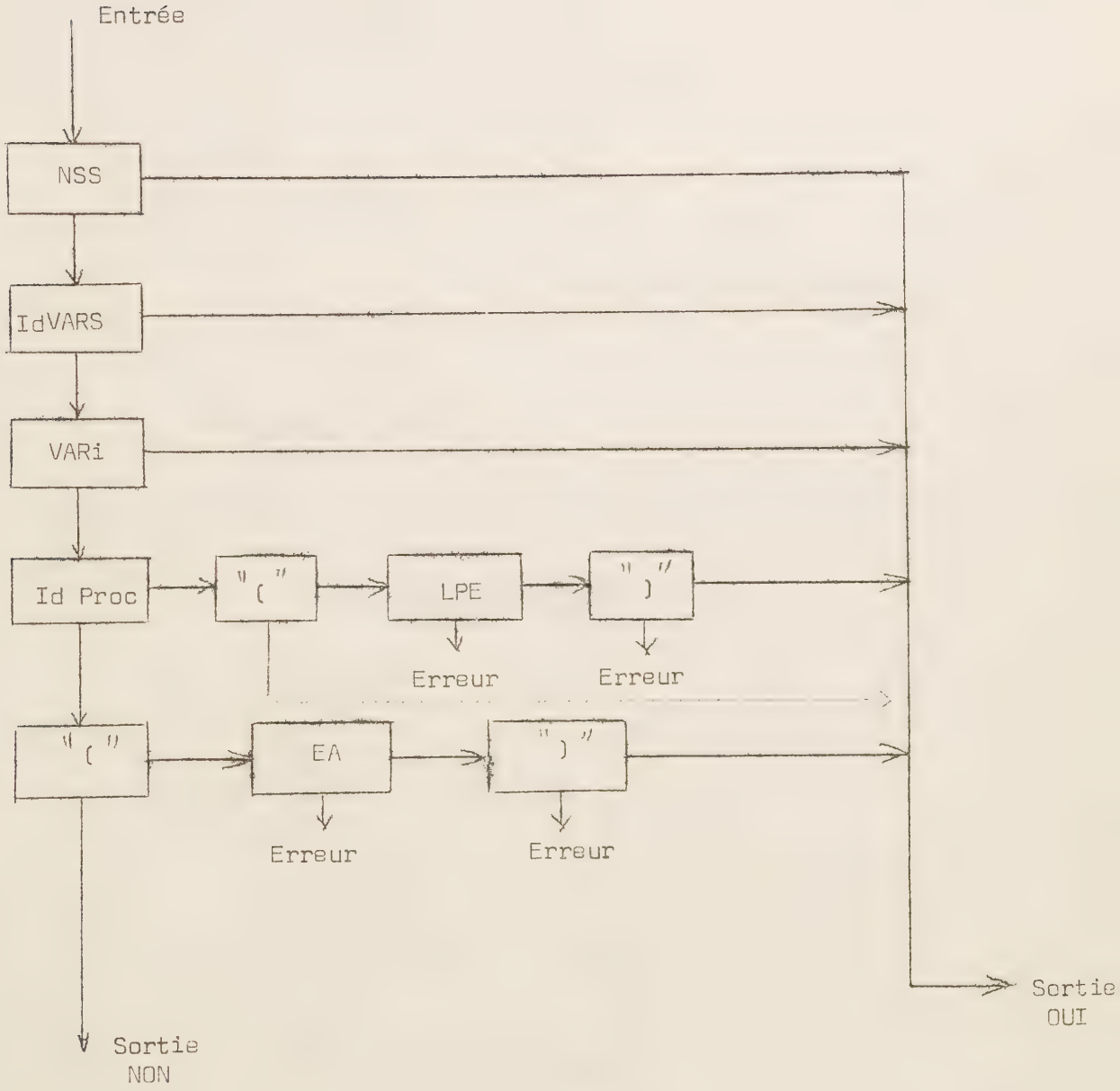
TERME ARITHMETIQUE



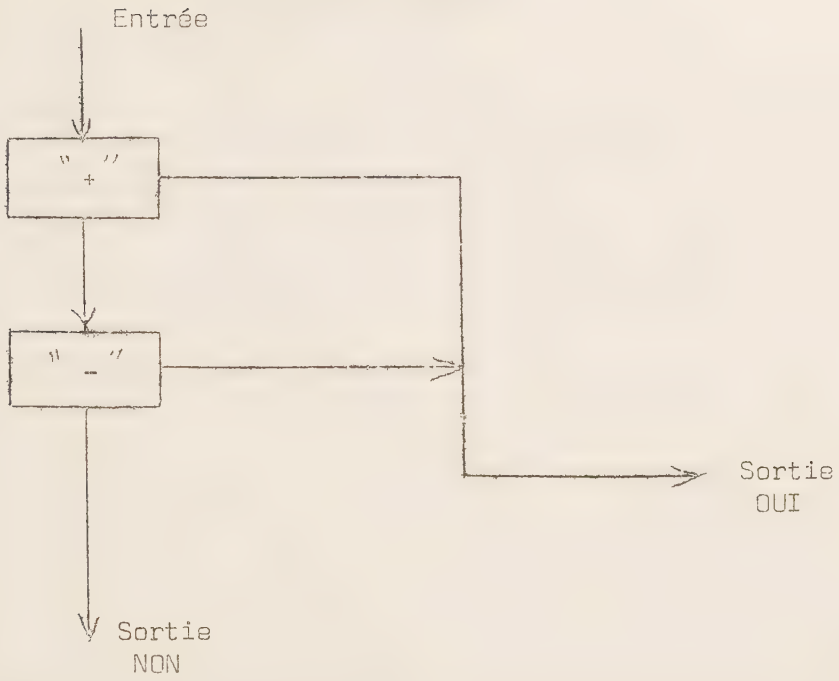
FACTEUR ARITHMETIQUE



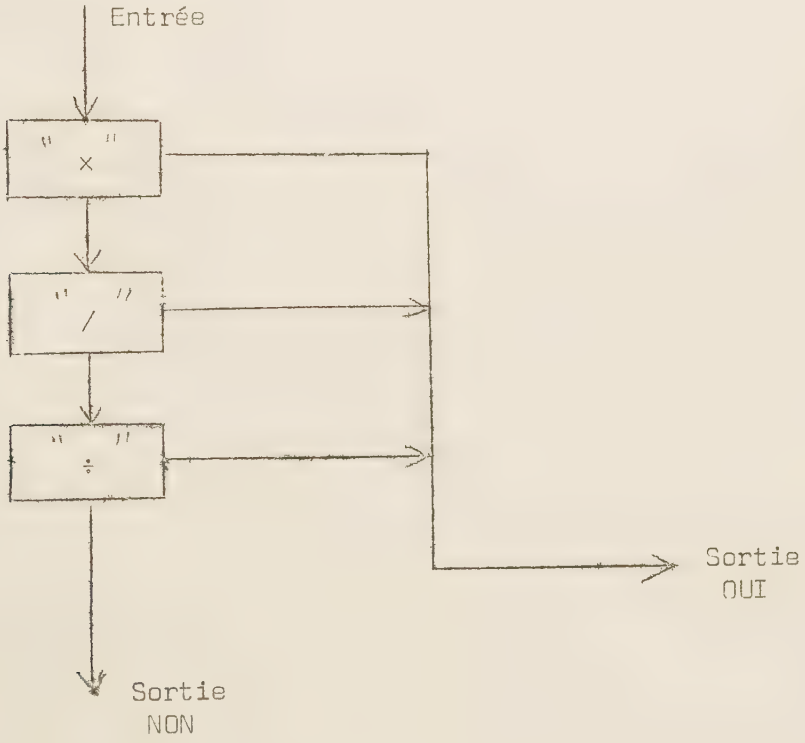
PRIMAIRE ARITHMETIQUE



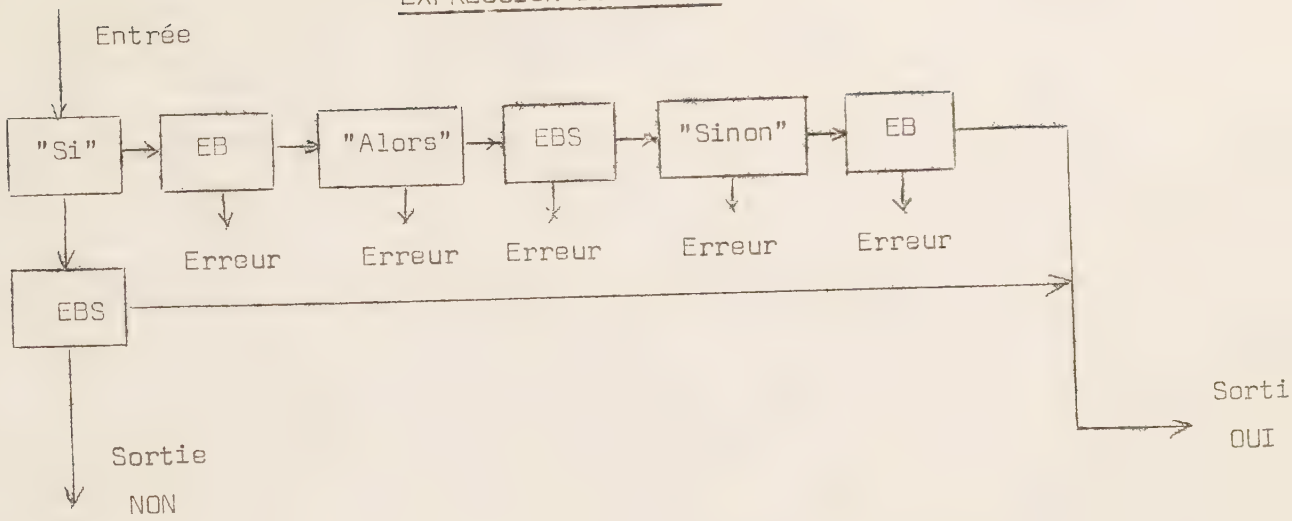
OPERATEUR ADDITIF



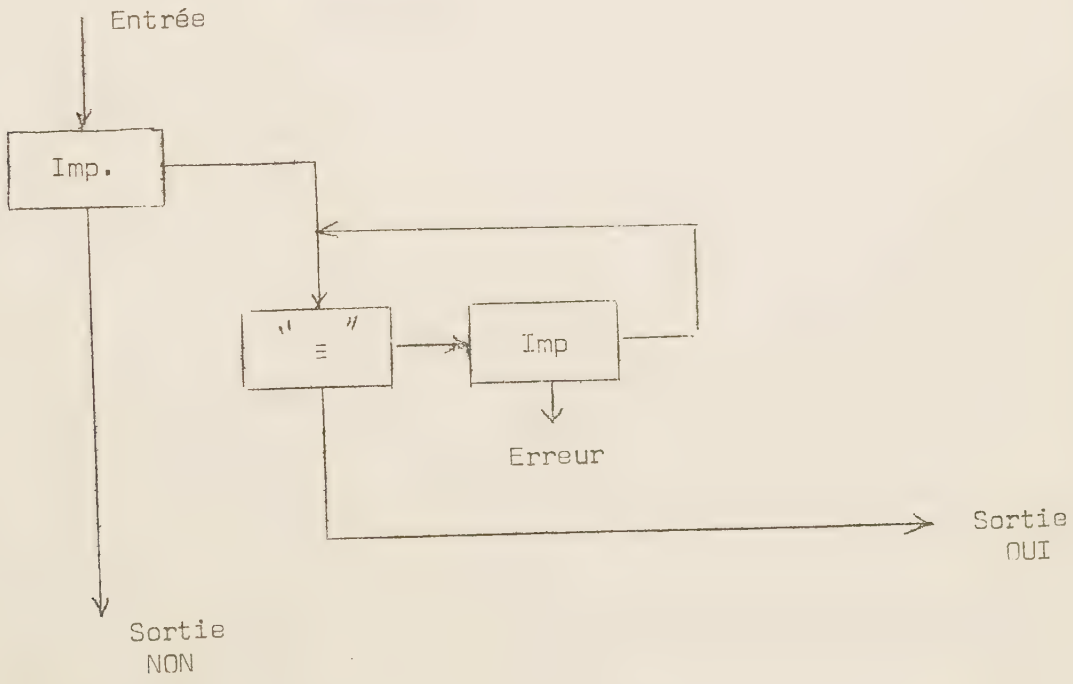
OPERATEUR MULTIPLICATIF



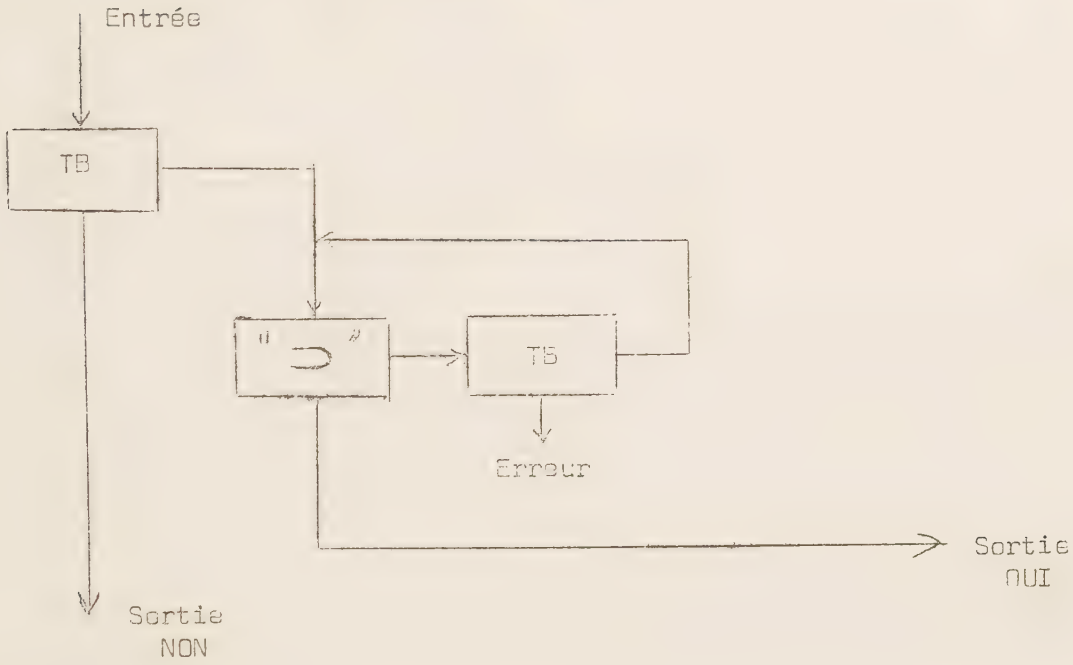
EXPRESSION BOOLEENNE



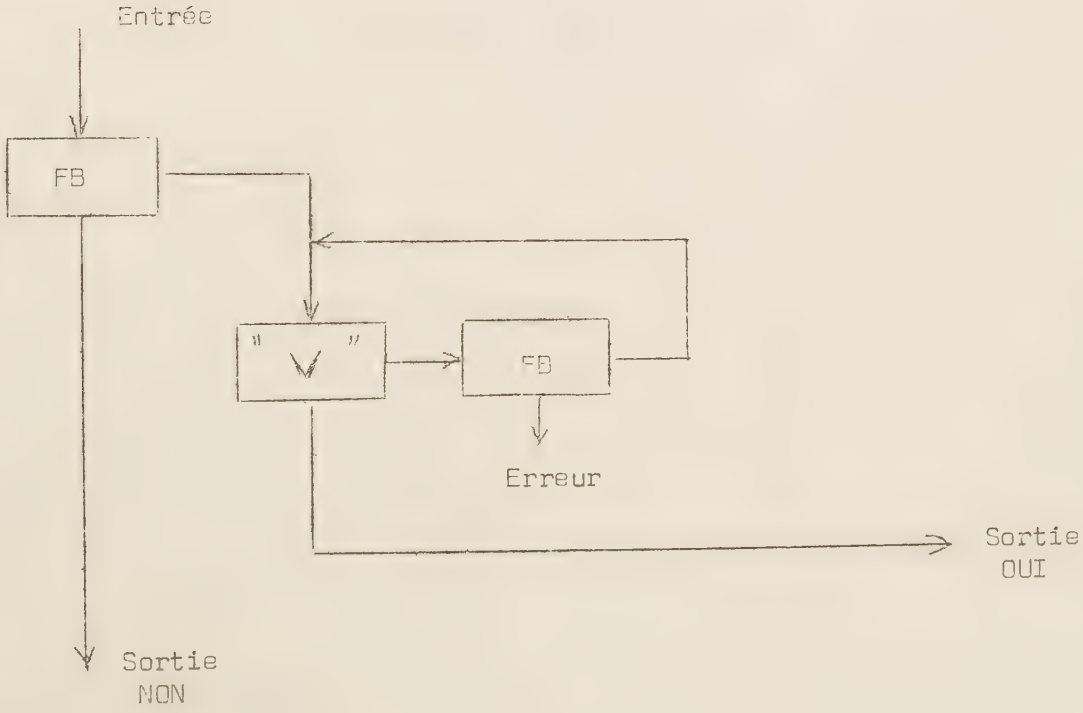
EXPRESSION BOOLEENNE SIMPLE



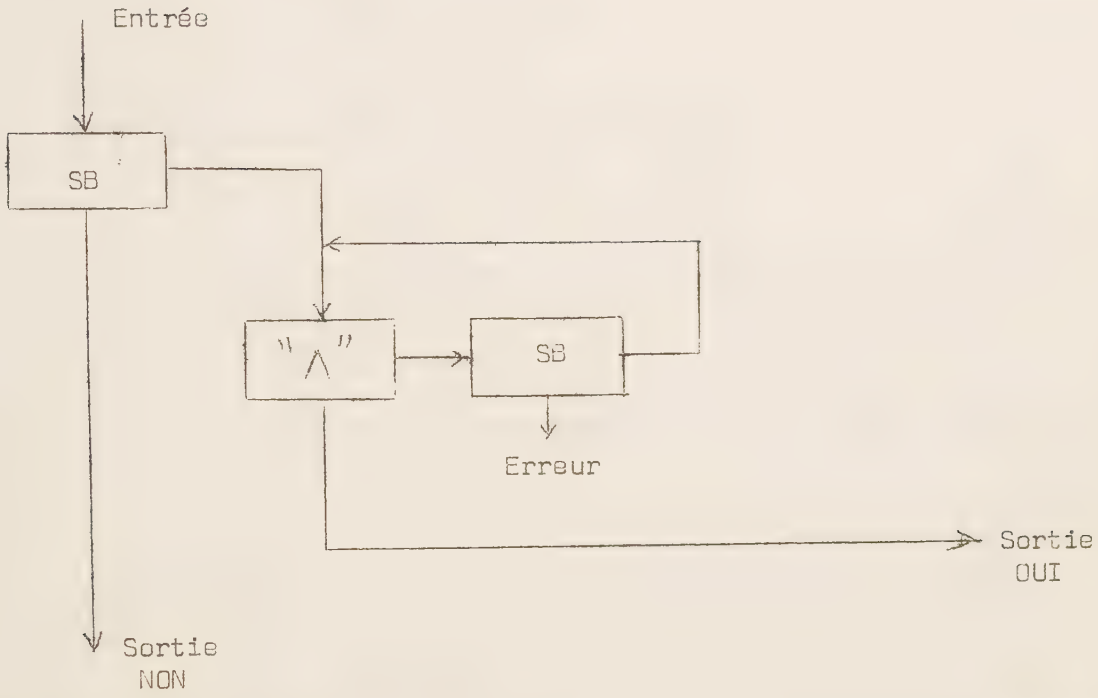
IMPLICATION



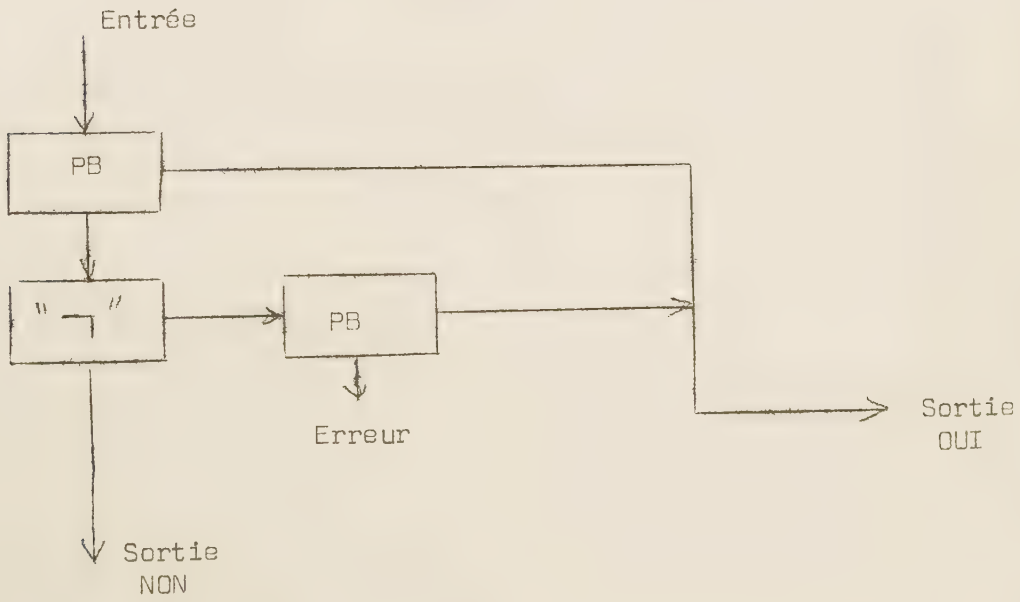
TERME BOOLEEN



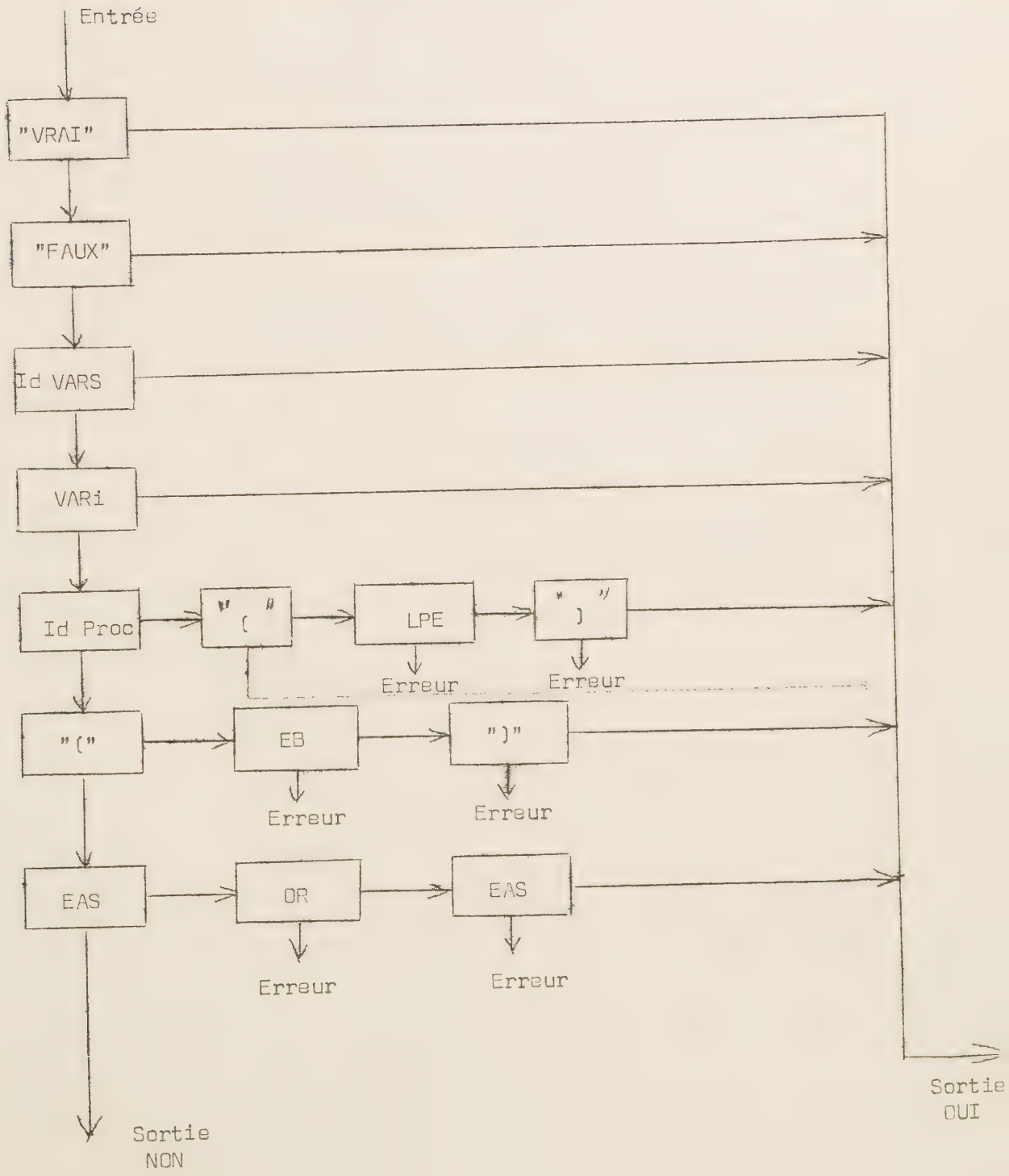
FACTEUR BOOLEEN



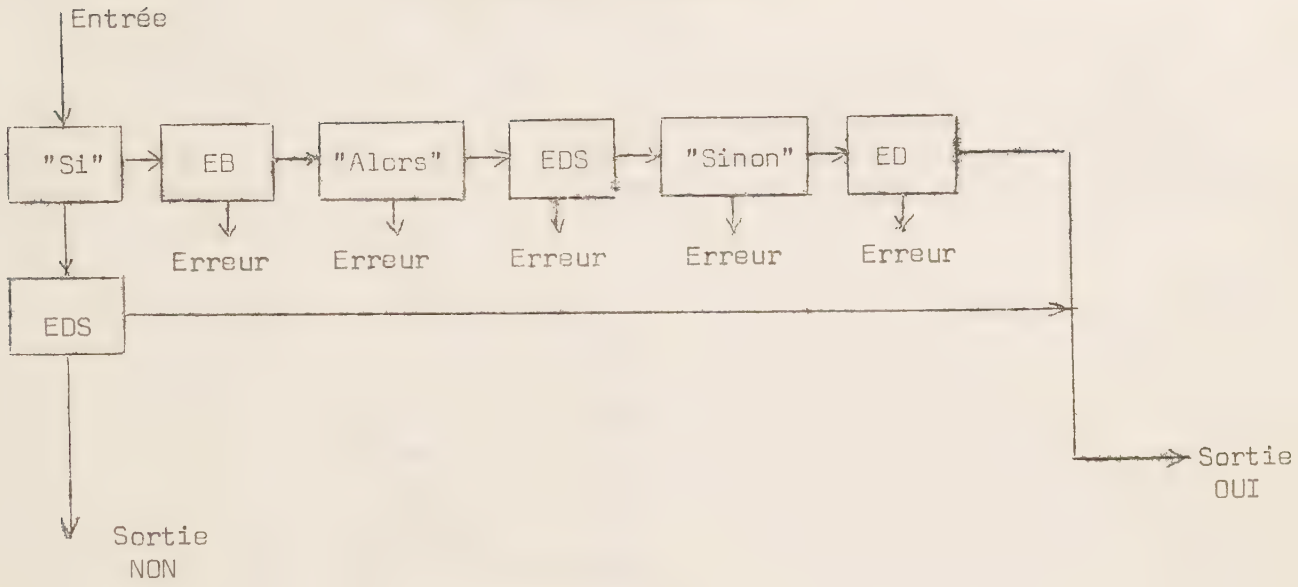
SECONDAIRE BOOLEEN



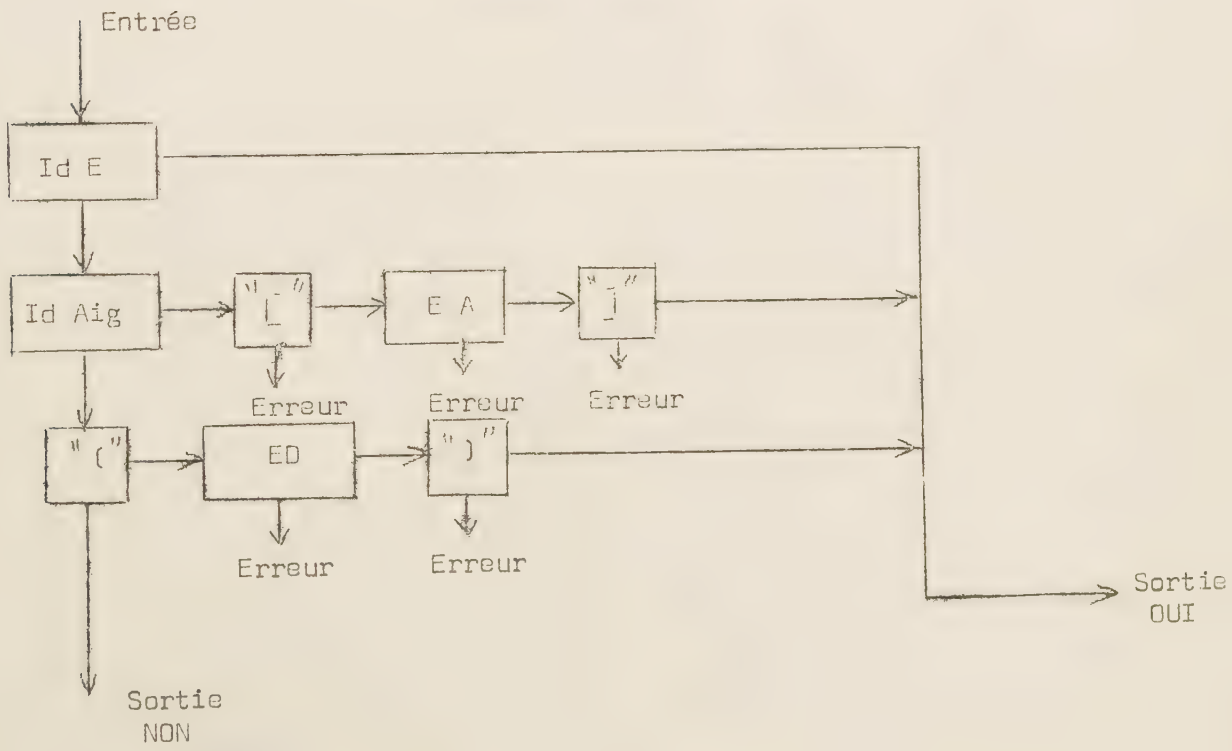
PRIMAIRE BOOLEEN



EXPRESSION DE DESIGNATION

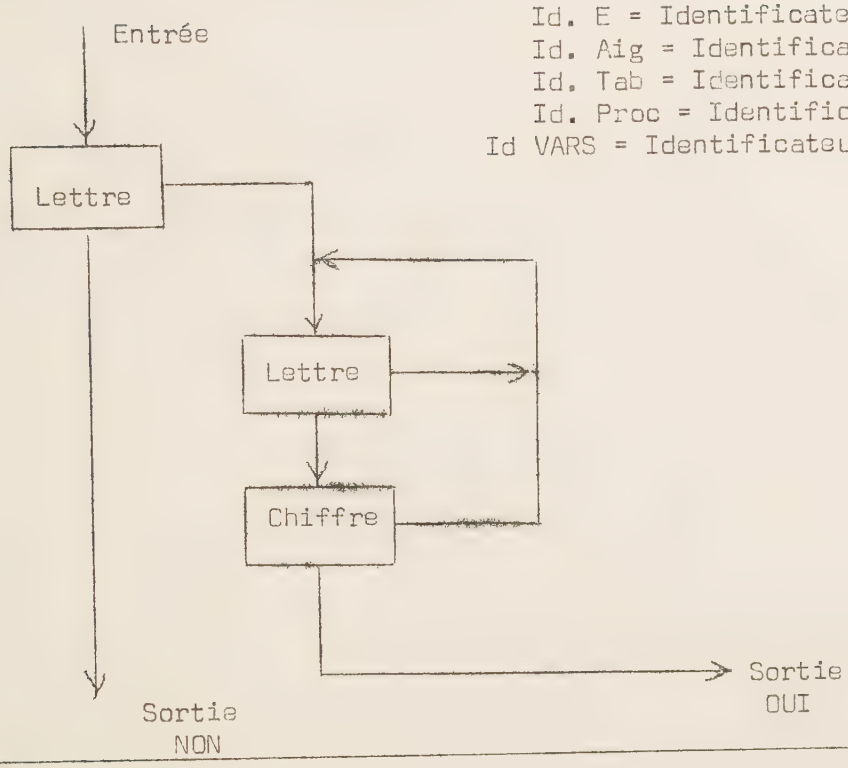


EXPRESSION DE DESIGNATION SIMPLE

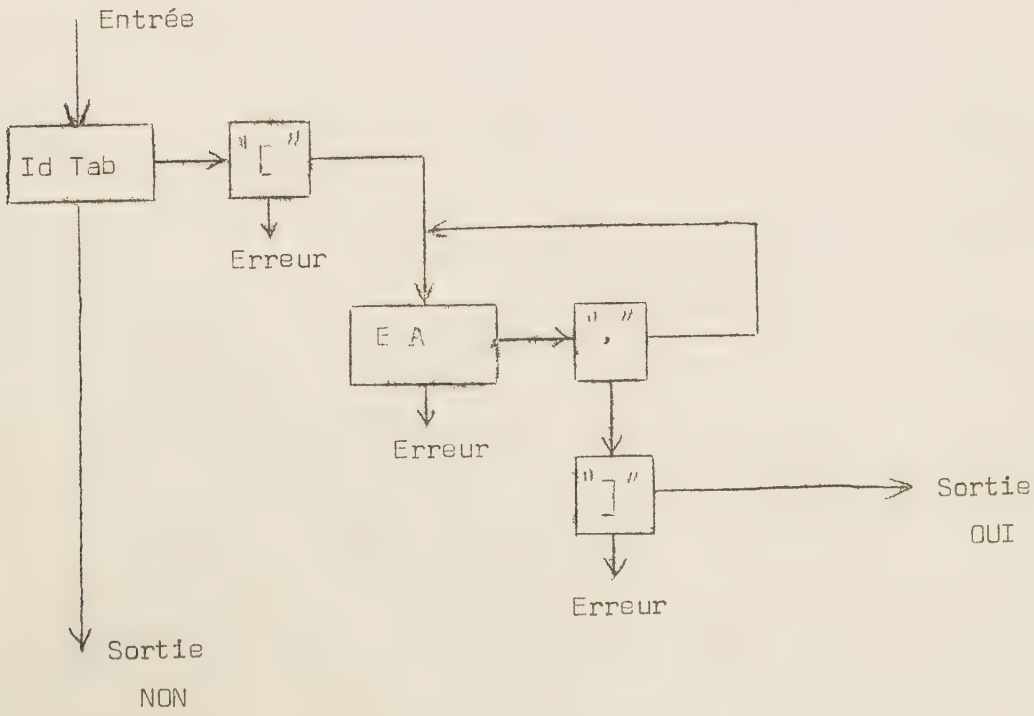


IDENTIFICATEUR

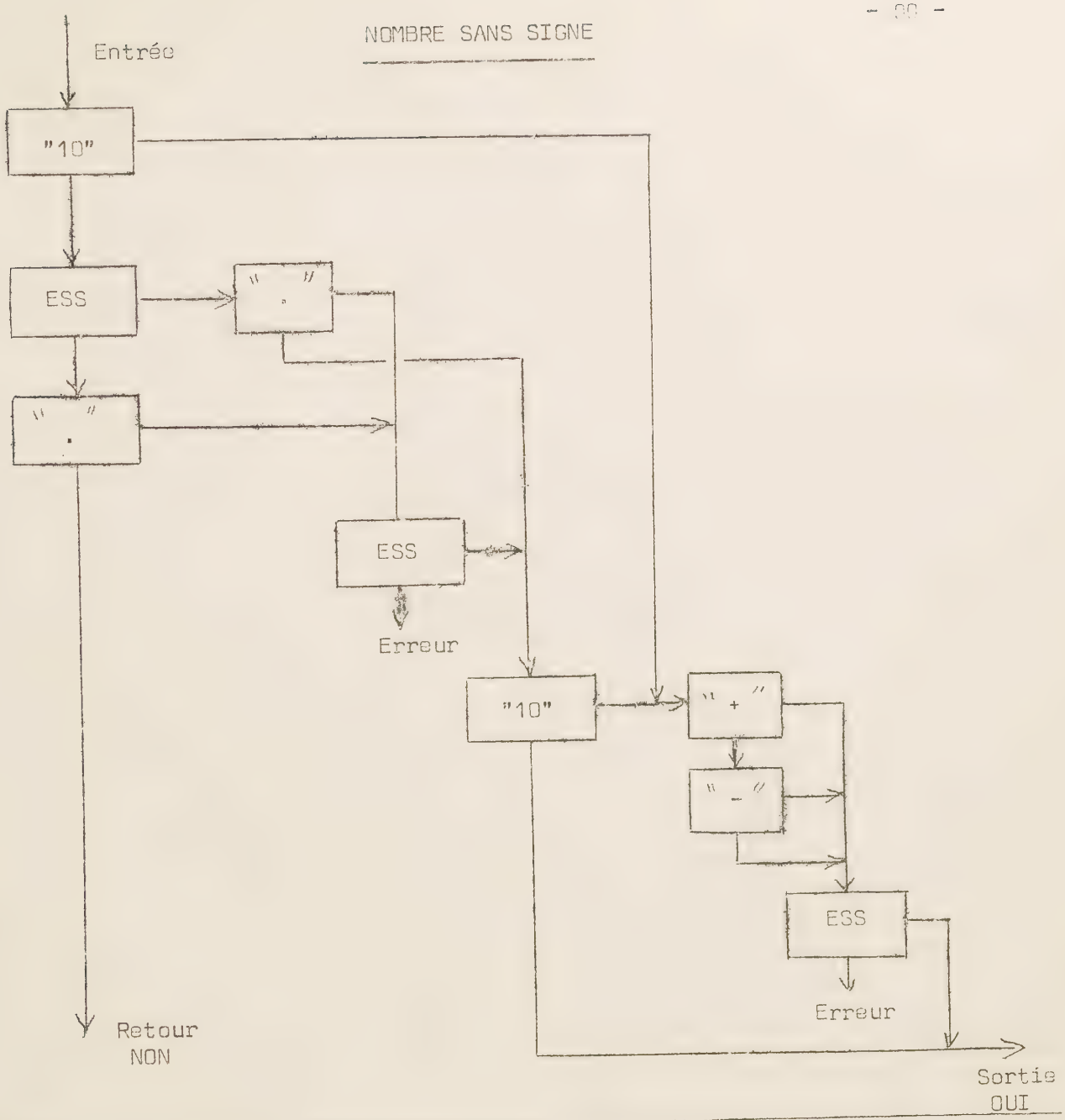
Id. E = Identificateur d'étiquette
Id. Aig = Identificateur d'aiguillage
Id. Tab = Identificateur de tableau
Id. Proc = Identificateur de procédure
Id VARS = Identificateur de variable simple



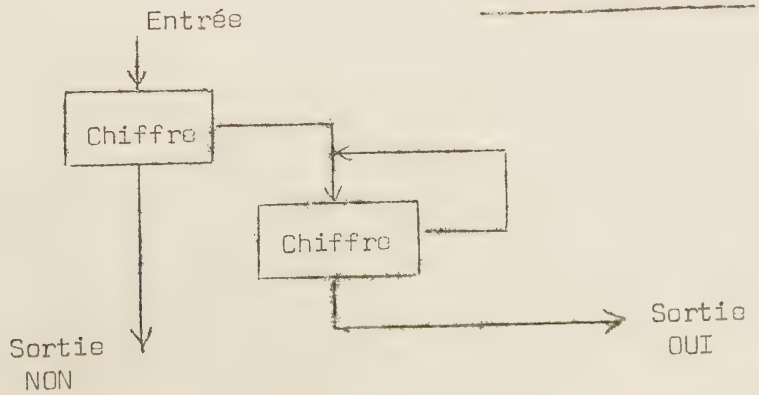
VARIABLE INDICEE



NOMBRE SANS SIGNE



ENTIER SANS SIGNE



TROISIEME PARTIE

TRADUCTION D'UN LANGAGE

LE PROBLEME DE LA TRADUCTION

Après avoir décrit un automate capable de reconnaître qu'un texte est écrit dans un langage (exemple : ALGOL), le problème de la traduction est de construire un autre automate capable de traduire ce texte dans un langage approprié au calculateur, ces deux textes étant, bien entendu, sémantiquement équivalents.

Le deuxième automate à constuire peut être appelé automate de "génération"

Le traducteur est alors l'ensemble des deux automates :

- l'un de reconnaissance d'un mot ALGOL
- l'autre de génération de ce mot en langage machine

La traduction peut se faire de différentes façons :

- soit globale, c'est-à- dire mot à mot
- soit élément libre par élément libre
- soit sous-mot par sous-mot, traduisible sémantiquement

C'est ce dernier cas que nous avons utilisé sur PB 250. Les parties de l'automate générateur sont alors imbriquées dans celles de l'automate de reconnaissance.

Ces parties de l'automate générateur, n'ont plus alors qu'à construire des séquences de programme en langage machine correspondantes à des sous-mots reconnus corrects, c'est-à-dire fabriquer des instructions machine et les placer dans des éléments de mémoire consécutifs.

Le problème revient alors à définir les équivalents sémantiques en langage machine des sous-mots d'un mot du langage tel qu'ALGOL.

La technique utilisée est celle des piles (automates à piles) (cf 4, 6, 7, 8) ,et qui peut être résumé comme suit :

Tant qu'un sous-mot n'a pas été complètement analysé, les éléments libres le constituant sont placés dans une pile de mémoire (ou plusieurs). Quand l'analyse du sous-mot est terminée, la partie de l'automate générateur fabrique à partir de celui-ci la séquence machine équivalente et efface ensuite le sous-mot de la pile.

On dit que l'on empile quand l'on place les éléments dans la pile et que l'on désempile quand on efface un ou plusieurs éléments, bien entendu seuls le ou les derniers éléments empilés sont accessibles. Nous ne pourrons pas entrer dans tous les détails de la traduction, il suffira de se rapporter au programme traducteur lui-même.

Langage machine PB 250 utilisé

Nota :

Nous n'utiliserons pas pour la description du langage machine de codes particuliers au calculateur (codes types numériques, mnémotechniques,etc ... tels que LDA, LDC, TRU, etc ...) ; ceci afin de faciliter la compréhension, le code machine n'ajoutant rien, seule sa signification étant importante.

Le programme résultant, traduction d'un programme ALGOL, est constitué d'une suite d'instructions machine et de tables où sont mémorisés les adresses de sauts, les constantes et les vecteurs de renseignements de tableaux.

Le programme machine résultant, proprement dit, comprend des instructions exécutables directement et des appels de sous-programmes.

Ces derniers sont nécessaires pour les calcule en "virgule flottante" (dans le cas PB 250) , les fonctions standards, etc ..., mais aussi facilitent la compilation dans la plupart des cas.

Un appel de sous-programme se fait toujours sous la même forme, seule. les adresses, le nombre des paramètres et les adresses des sous-programmes varient.

La forme générale d'un appel de sous-programme est la suivante :

- transmission au sous-programme des paramètres par des registres arithmétiques ou des mémoires intermédiaires
- saut inconditionnel vers le sous-programme

Les paramètres sont en général les opérandes et l'instruction de retour.

Les renseignements intervenant dans la confection d'un programme en langage machine sont :

- les codes opérations
- les adresses mémoire des données et des sauts

Quatre tâches principales reviennent au programme générateur :

- Affectation d'adresses aux identificateurs de variables, toutes les fois que cela est possible
- Repérage des adresses de sauts pendant la traduction
- Génération proprement dite du programme résultant
- Empilage et désempilage des sous-mots pendant la traduction

Pour la description des parties del'automate générateur, on aura recours en général aux organigrammes où seront représentées les parties de l'automate de reconnaissance et celles du générateur.

Traduction des Expressions

En ALGOL, on a trois types d'expressions :

- les expressions arithmétiques
- les expressions booléennes
- les expressions de désignation

Le principe de traduction est le même pour ces trois types d'expressions, seules des différences apparaissent au niveau de l'adressage des opérandes et de l'adresse des sous-programmes opérateurs.

Le programme en langage machine obtenu est la transcription des expressions en notation post fixée dans laquelle apparaissent tous les résultats intermédiaires.

La traduction s'effectue par la technique des piles (métbode type : Bauer et Samelson).

On utilise en général deux piles, pour faciliter la programmation (une seule pile suffirait) :

- une pile opérandes
- une pile opérateurs

En pile opérandes, on stocke les adresses de variables ou de mémoires de manoeuvres.

En pile opérateurs, on stocke les adresses des sous-programmes opérateurs.

Remarque

En vue de simplifier les organigrammes, on donne ci-après, le fonctionnement d'une partie d'automate de génération type "Expression"

Générer :

1 - Appel du sous-programme correspondant à l'opérateur en pile "opérateur", avec comme paramètres les deux opérandes en début de pile " opérandes"

2 - Transmission du résultat en mémoire de manoeuvre

Empiler :

l'adresse de la mémoire de manoeuvre en pile opérandes

Cette partie d'automate est la même pour :

- une expression arithmétique simple
- un terme arithmétique
- un facteur arithmétique
- une expression booléenne simple
- une implication
- un terme booléen
- un facteur booléen
- un secondaire booléen
- une relation

EXPRESSION ARITHMETIQUE

Sémantique

Une expression arithmétique est une règle pour le calcul d'une valeur numérique.

Deux types d'expressions arithmétiques existent :

EAS : une expression arithmétique simple
EA : une expression arithmétique quelconque

Lorsqu'il s'agit d'expressions arithmétiques simples, on obtient la valeur en exécutant les opérations arithmétiques indiquées sur les valeurs effectives des primaires arithmétiques de l'expression.

Dans les expressions arithmétiques les plus générales comprenant des propositions si, une seule parmi diverses expressions arithmétiques simples est choisie en fonction des valeurs actuelles des expressions booléennes.

Ce choix peut être résumé par :

soit B : une expression booléenne

On a pour forme générale d'une expression arithmétique

"Si" B "alors" EAS "Sinon" EA

signifiant :

si B est vrai, calculer l'EAS
si B est faux, calculer l'E.A

La traduction d'une forme conditionnelle est la même dans toutes les expressions et les instructions conditionnelles. Seuls "Alors" et "Sinon" sont traduits, et la description sera faite dans le cas d'une expression booléenne.

Traduction d'une expression arithmétique

La principe de la traduction peut être illustré par un exemple :

soit à traduire l'expression arithmétique

(1) $F = a \uparrow b \uparrow c \times d \uparrow e$

signifiant :

$m_1 = a \uparrow b$
$m_2 = m_1 \uparrow c$
$m_3 = d \uparrow e$
$m_4 = m_2 \times m_3$
$F = m_4$

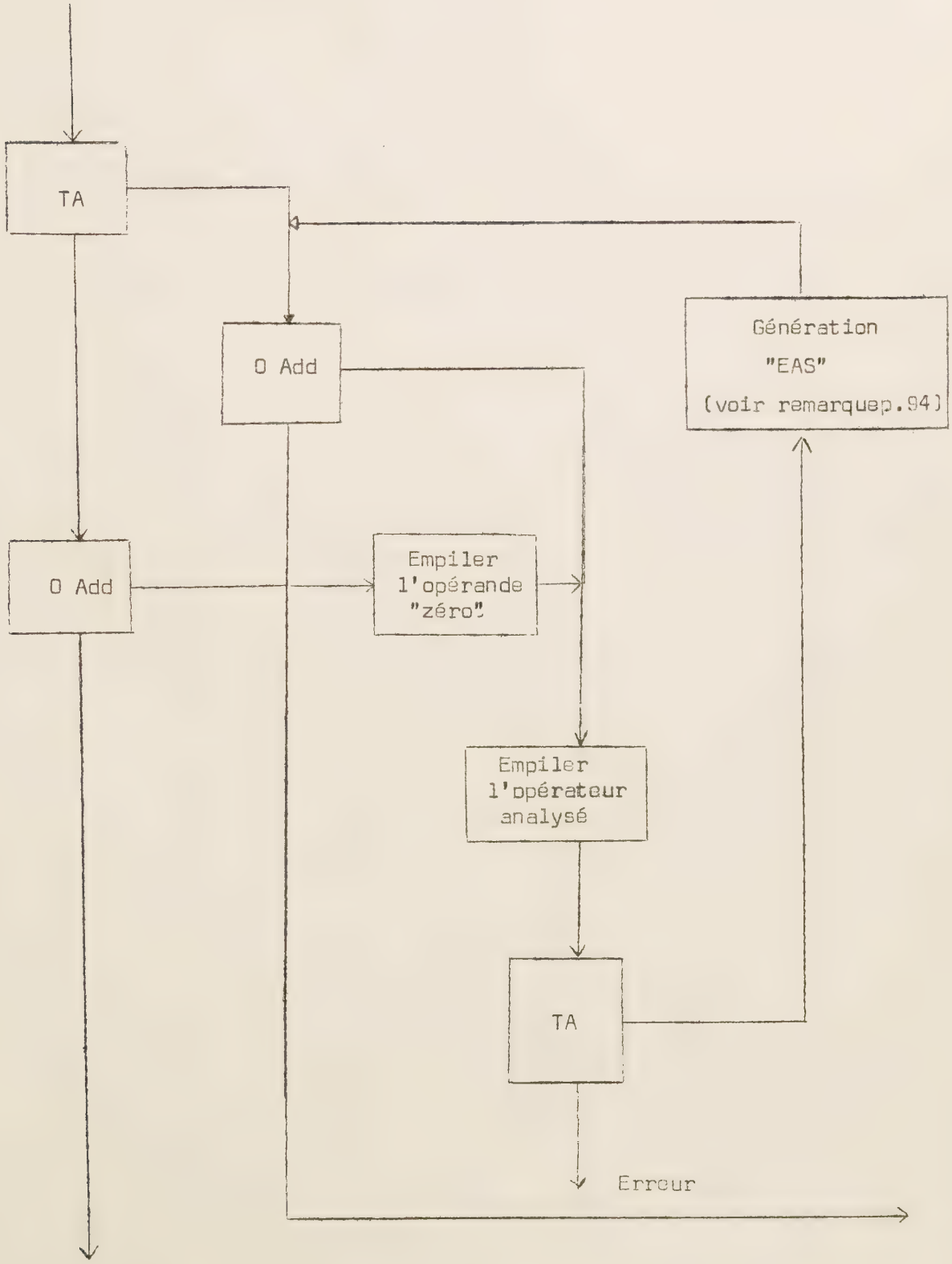
soit en notation post fixée :

$a\ b \uparrow m_1 =$
$m_1\ c \uparrow m_2 =$
$d\ e \uparrow m_3 =$
$m_2\ m_3 \times m_4 =$
$m_4\ F =$

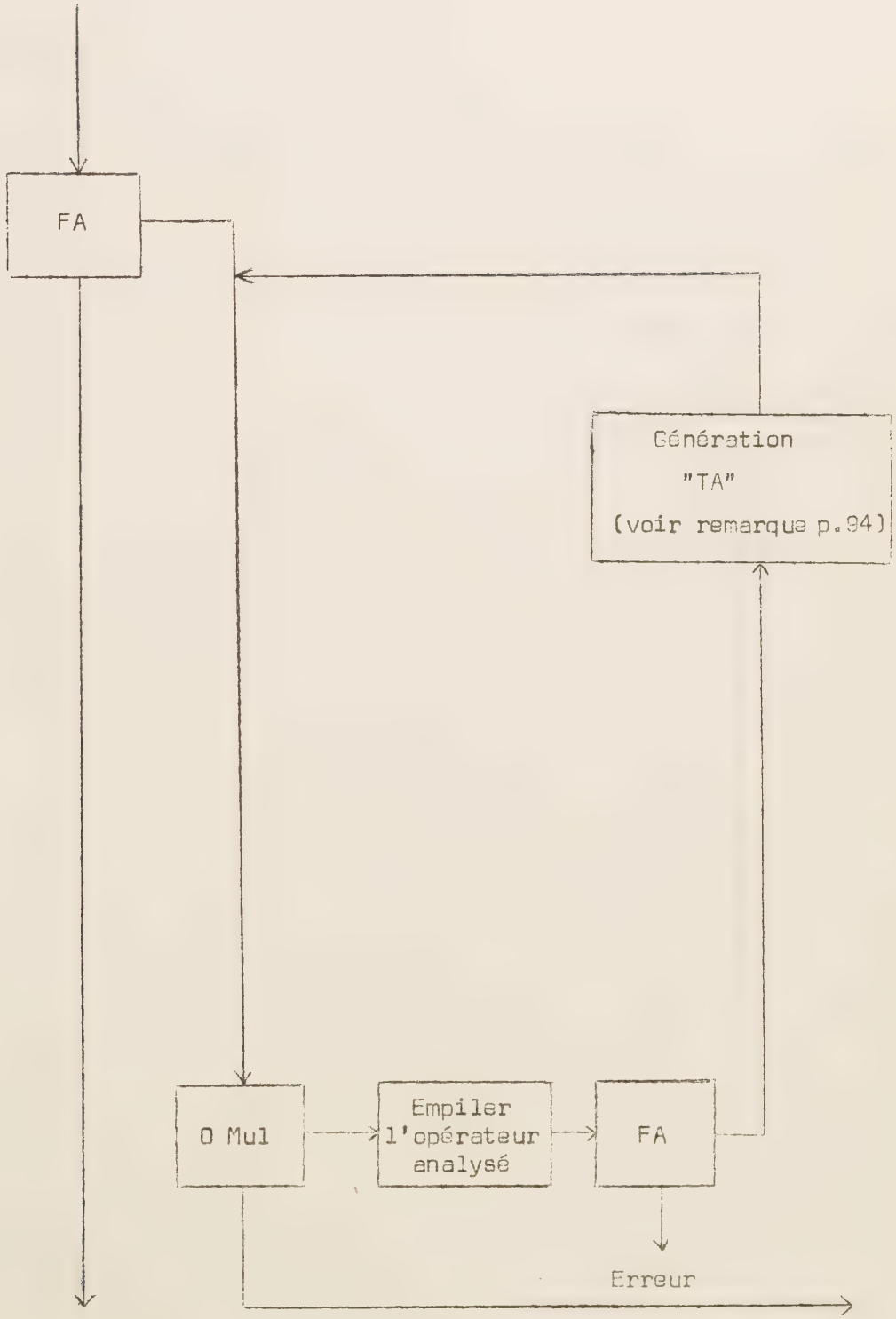
représentant, sous forme symbolique, le programme résultant en langage machine (m_1, m_2, m_3, m_4 : sont des mémoires de manoeuvre)

Les parties de l'automate générateur seront décrites sous forme d'organigramme

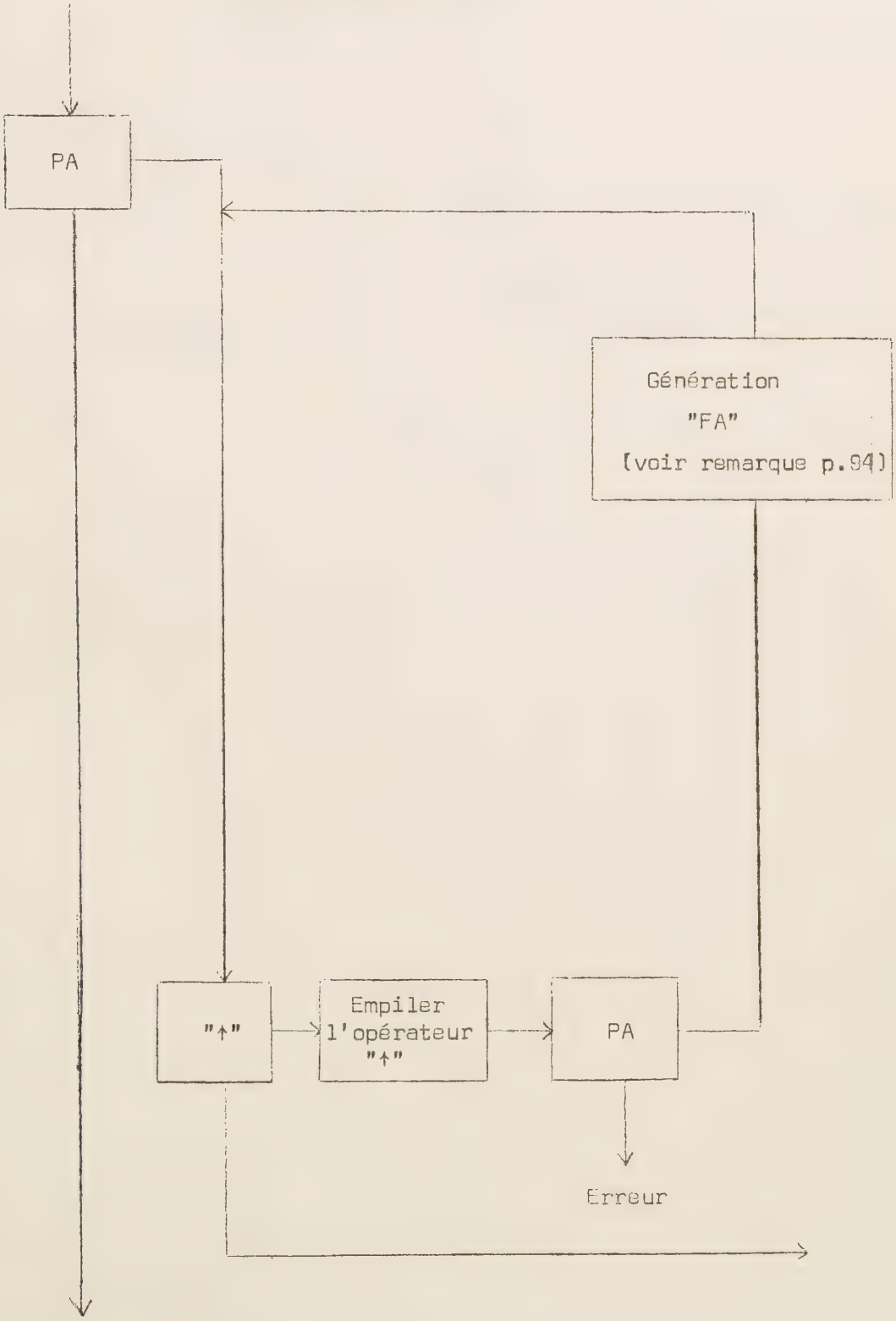
EXPRESSION ARITHMETIQUE SIMPLE



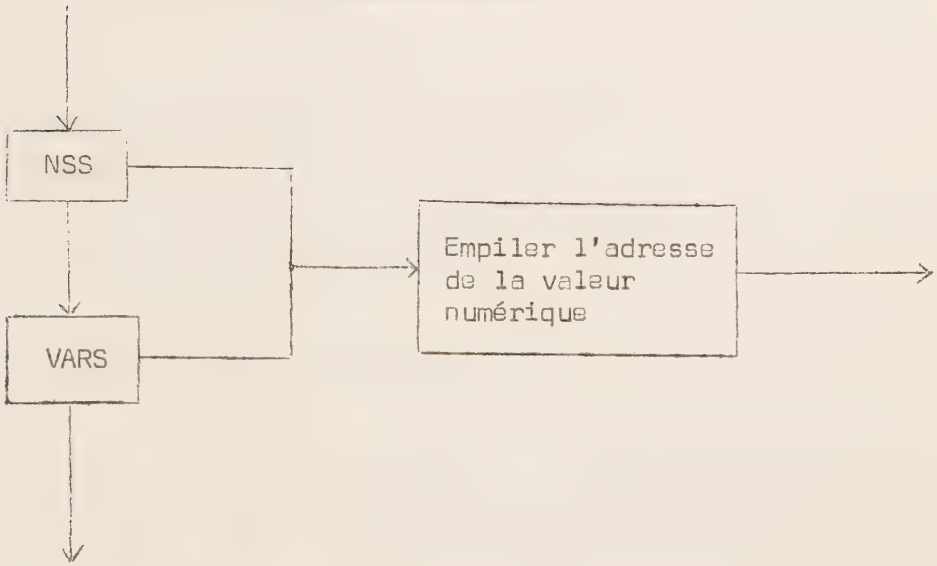
TERME ARITHMETIQUE



FACTEUR ARITHMETIQUE



PRIMAIRE ARITHMETIQUE



Nota :

La traduction d'une variable indicée sera anlysée à part. Après l'analyse d'une variable indicée, on trouve en pile opérandes une adresse de mémoire de manoeuvre.

EXPRESSION BOOLEENNE

Sémantique

Une expression booléenne est une règle pour le calcul d'une valeur logique.

Les principes du calcul sont analogues à ceux qui ont été donnés pour les expressions arithmétiques.

Traduction d'une expression booléenne

Généralités

Dans les calculateurs arithmétiques, l'existence d'instructions machine de rupture de séquence conditionnelle de la forme :

"Transfert si le contenu d'un registre est négatif"

conduit à adopter pour valeurs logiques :

VRAI : + 0

FAUX : - 0

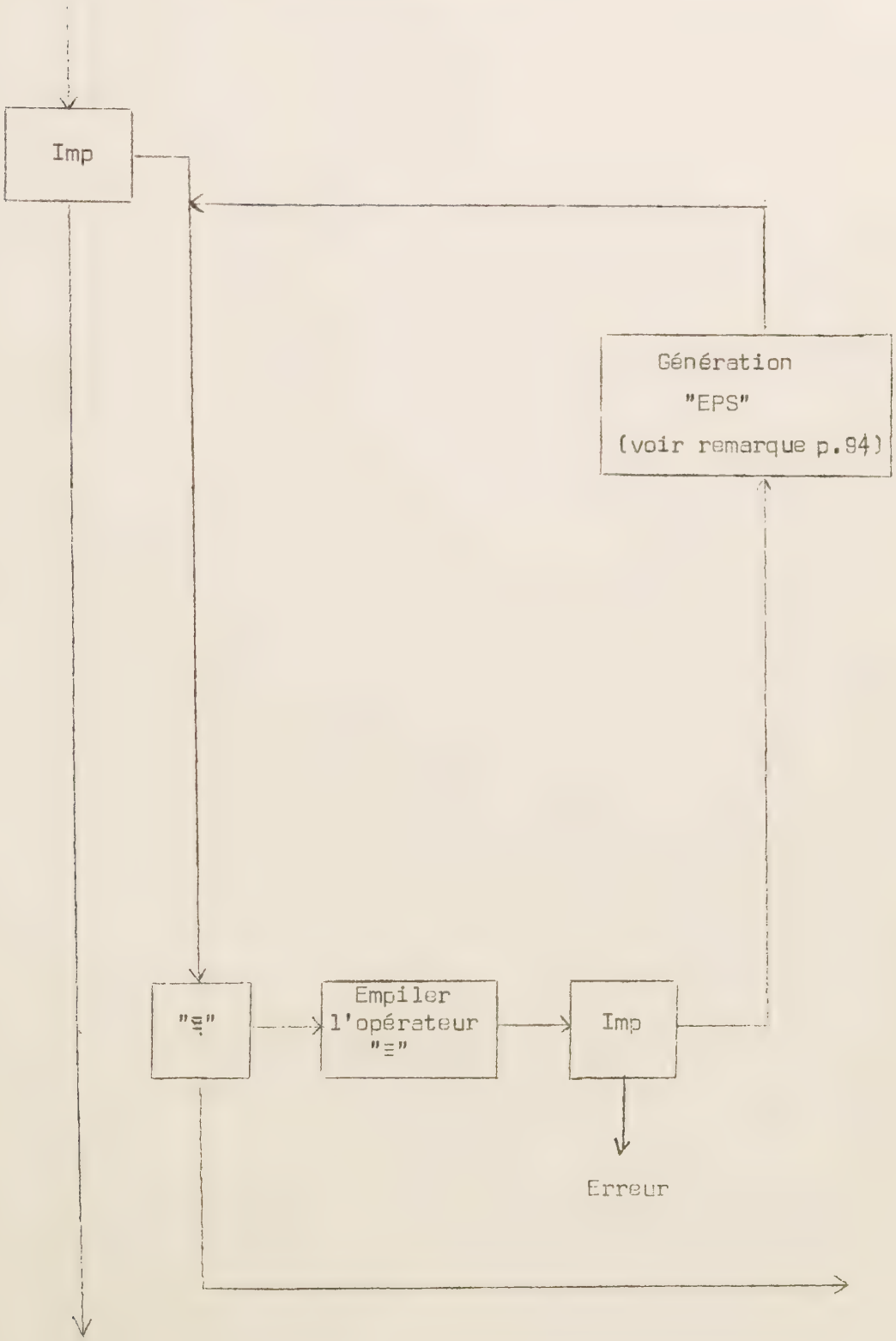
c'est-à-dire quel'on utilise une logique dite "négative"

Alors une seule instruction machine suffit pour tester une expression booléenne, et permettre l'aiguillage sur deux parties de programme différentes suivant la valeur de l'expression booléenne.

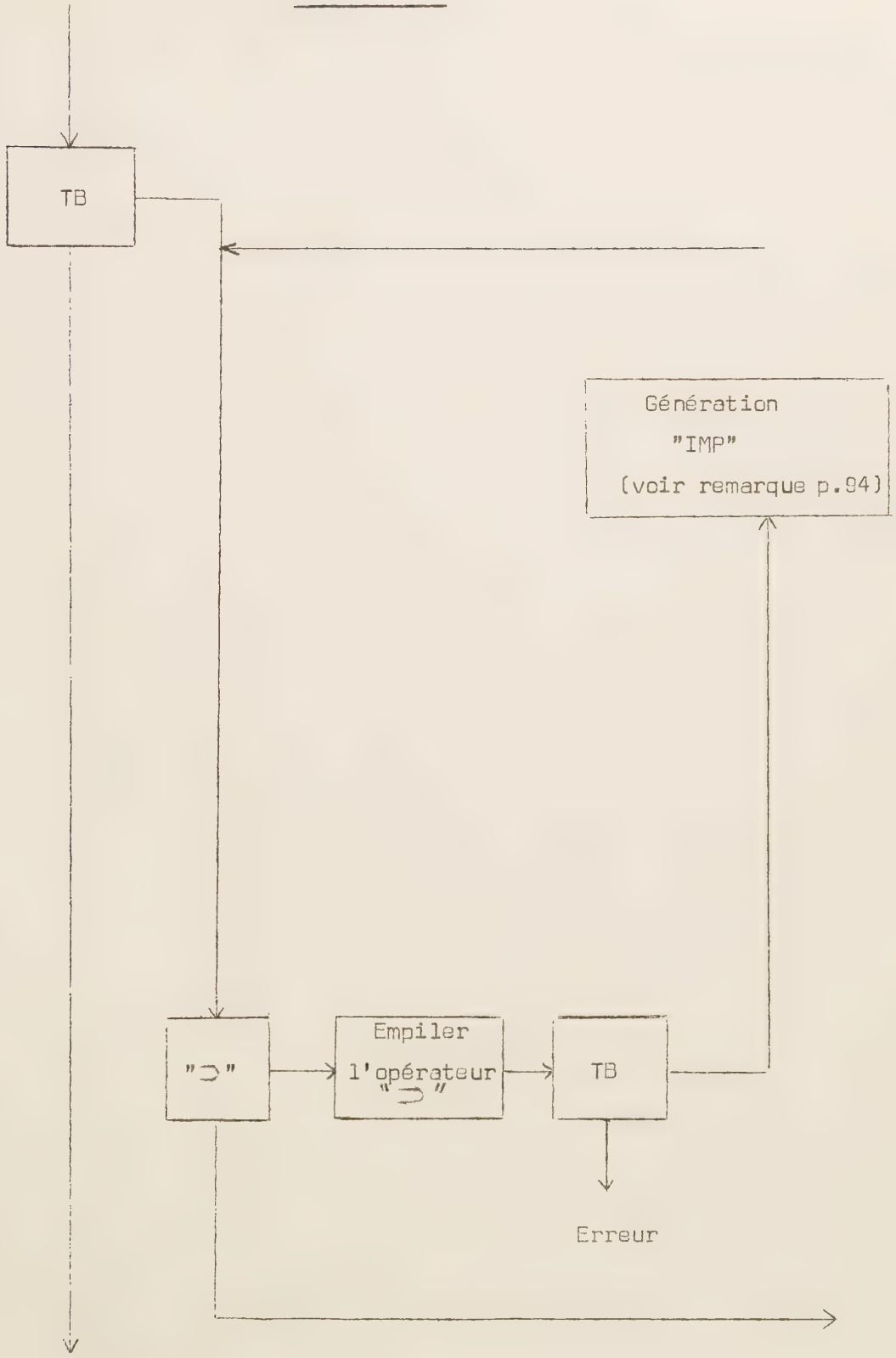
La traduction s'effectue de la même façon que celle d'une expression arithmétique, en utilisant les piles "opérandes" et "opérateurs"

On obtient les organigrammes suivants :

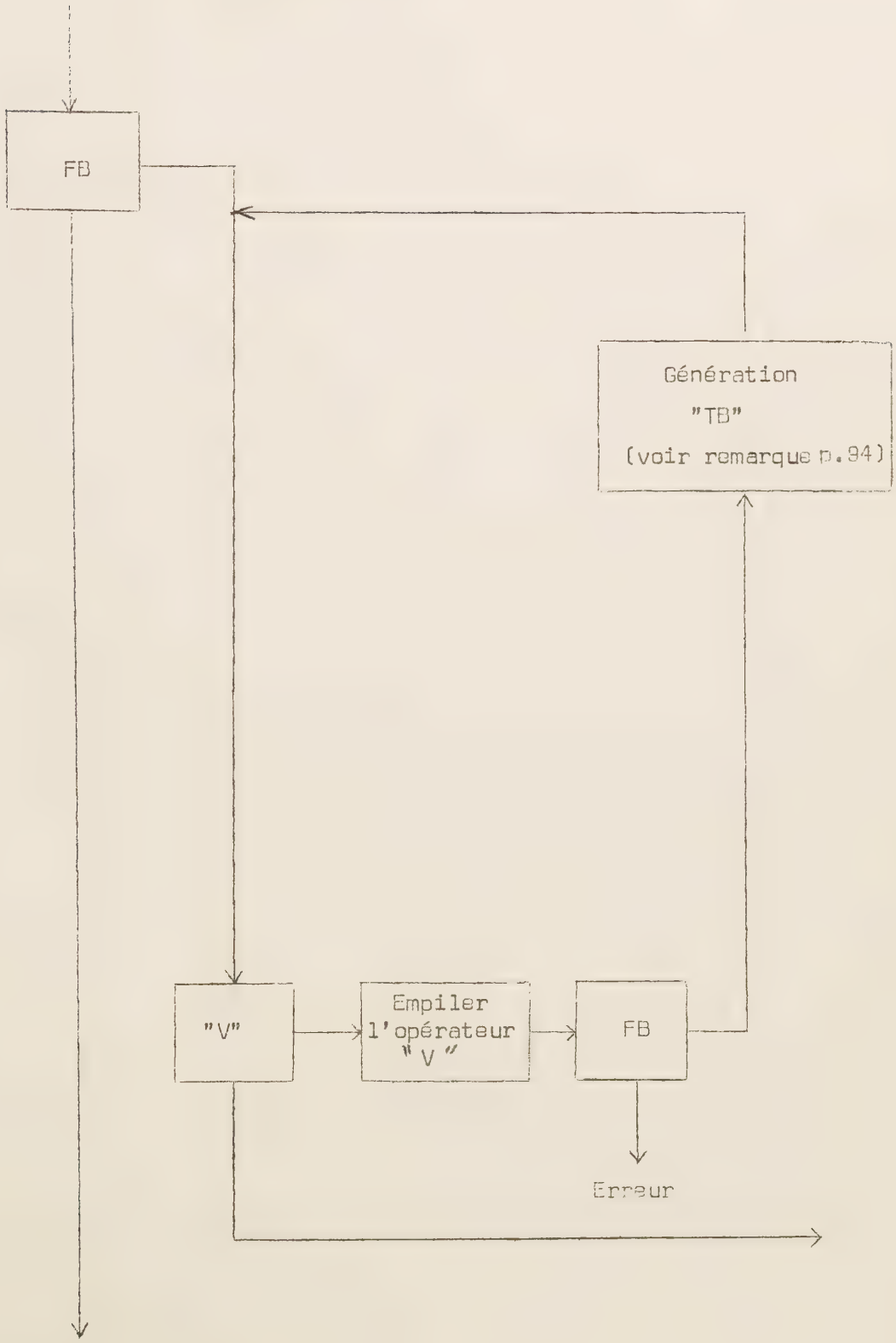
EXPRESSION BOOLEENNE SIMPLE



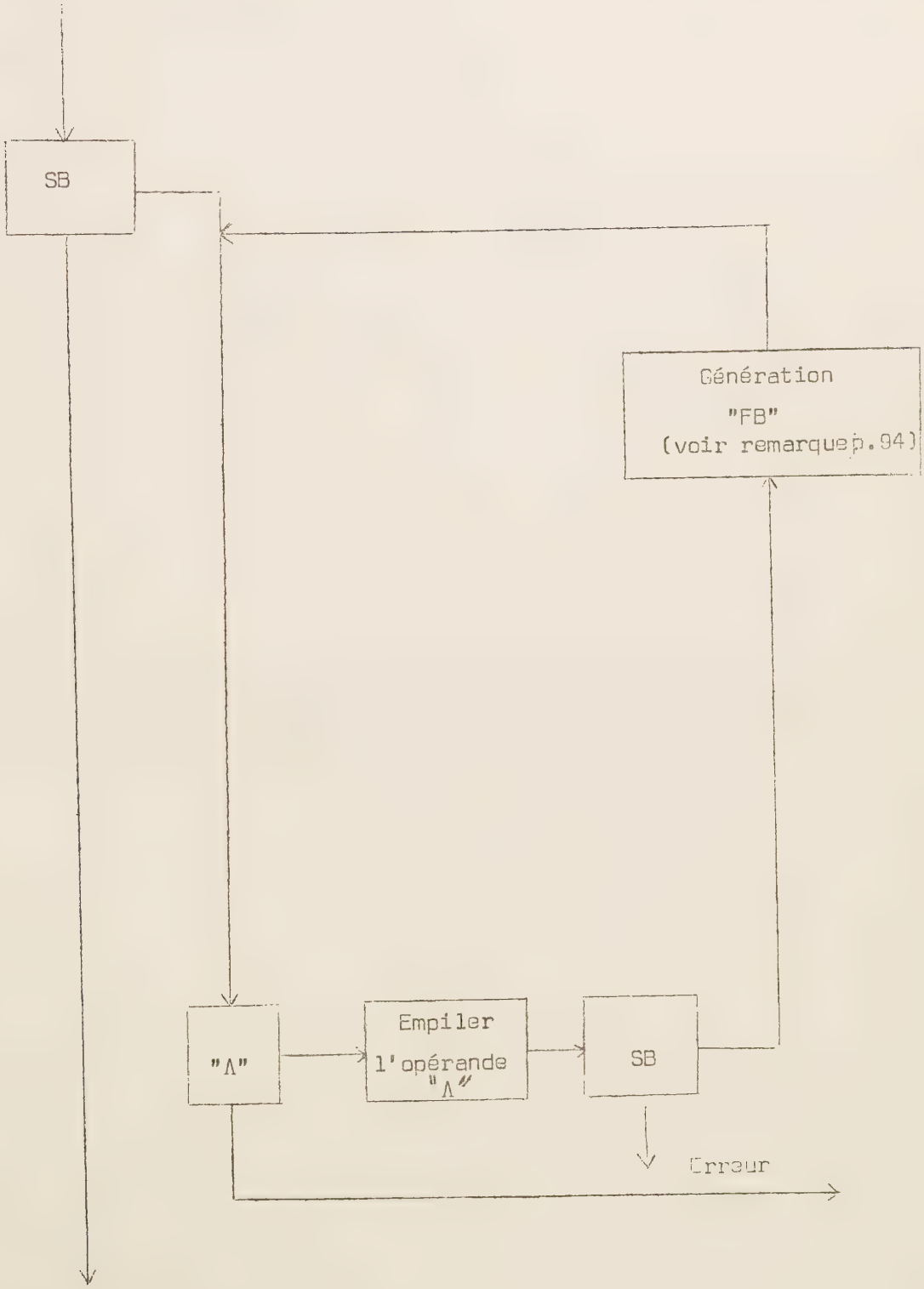
IMPLICATION



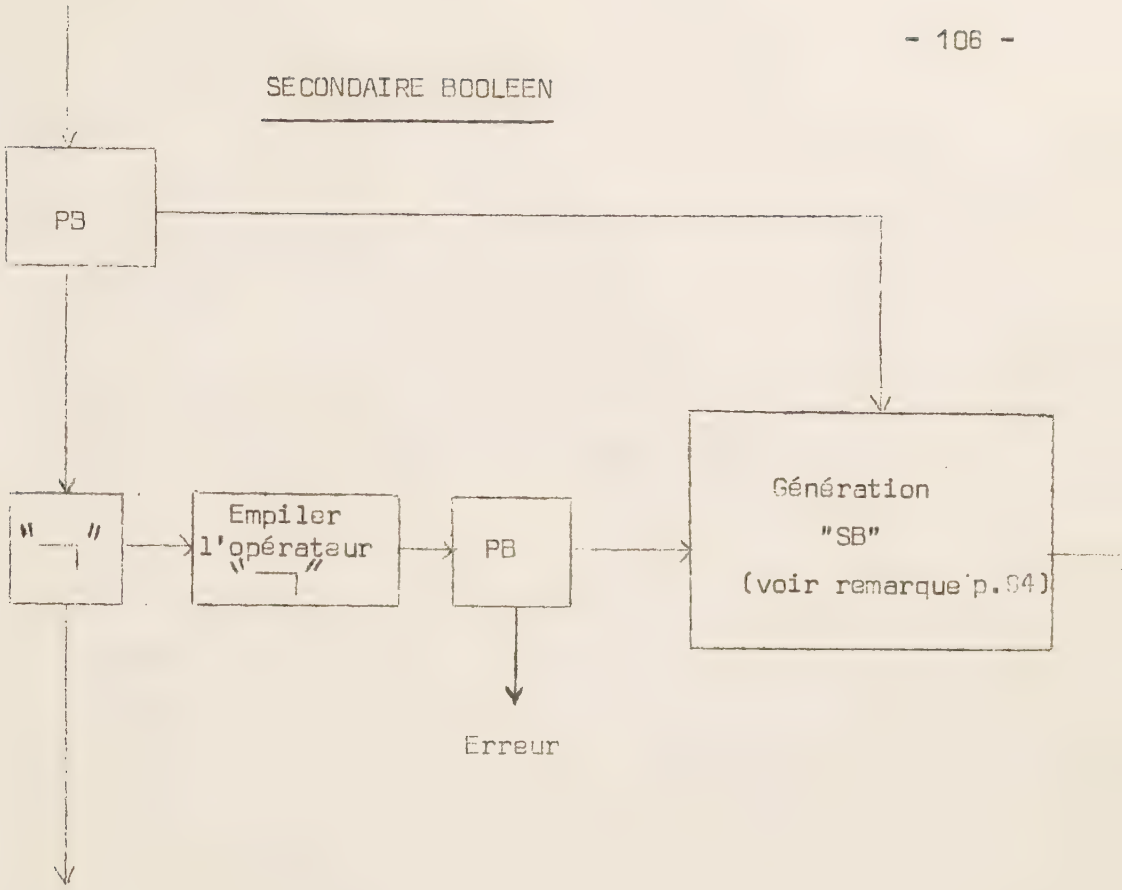
TERME BOOLEEN



FACTEUR BOOLEEN



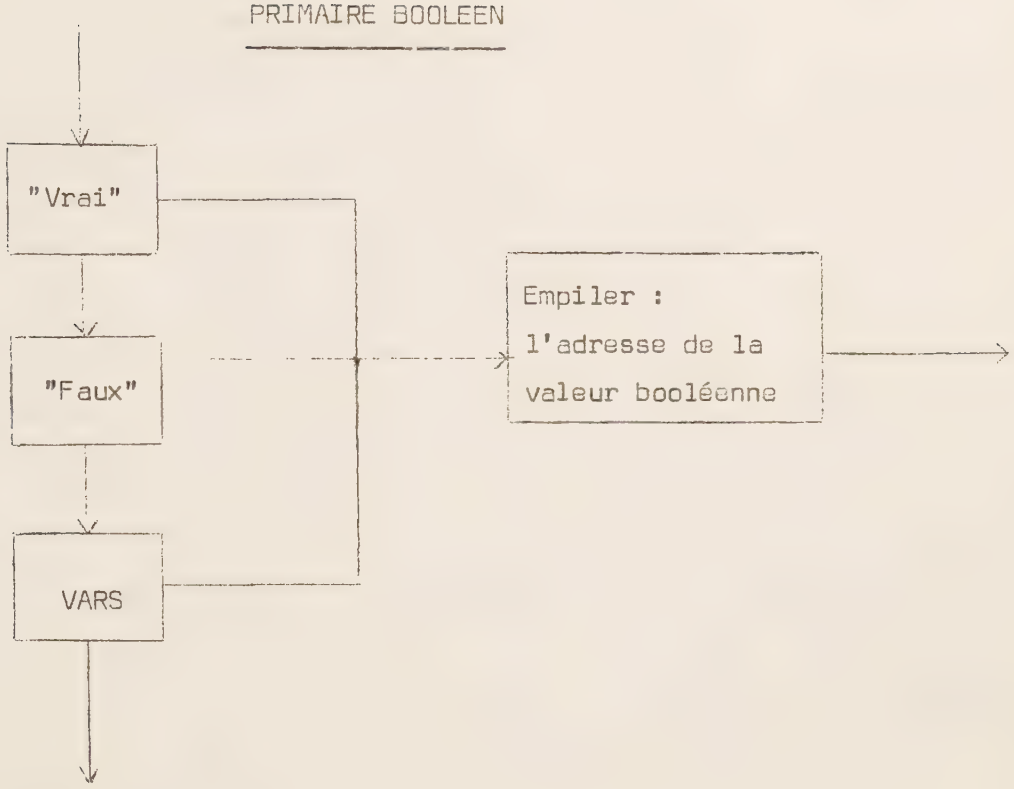
SECONDAIRE BOOLEEN



Nota :

Afin de simplifier le compilateur, on génère un appel de sous-programmes avec deux opérandes, mais le sous-programme réalisant (¬ . PB) , à l'exploitation n'utilise qu'un seul opérande.

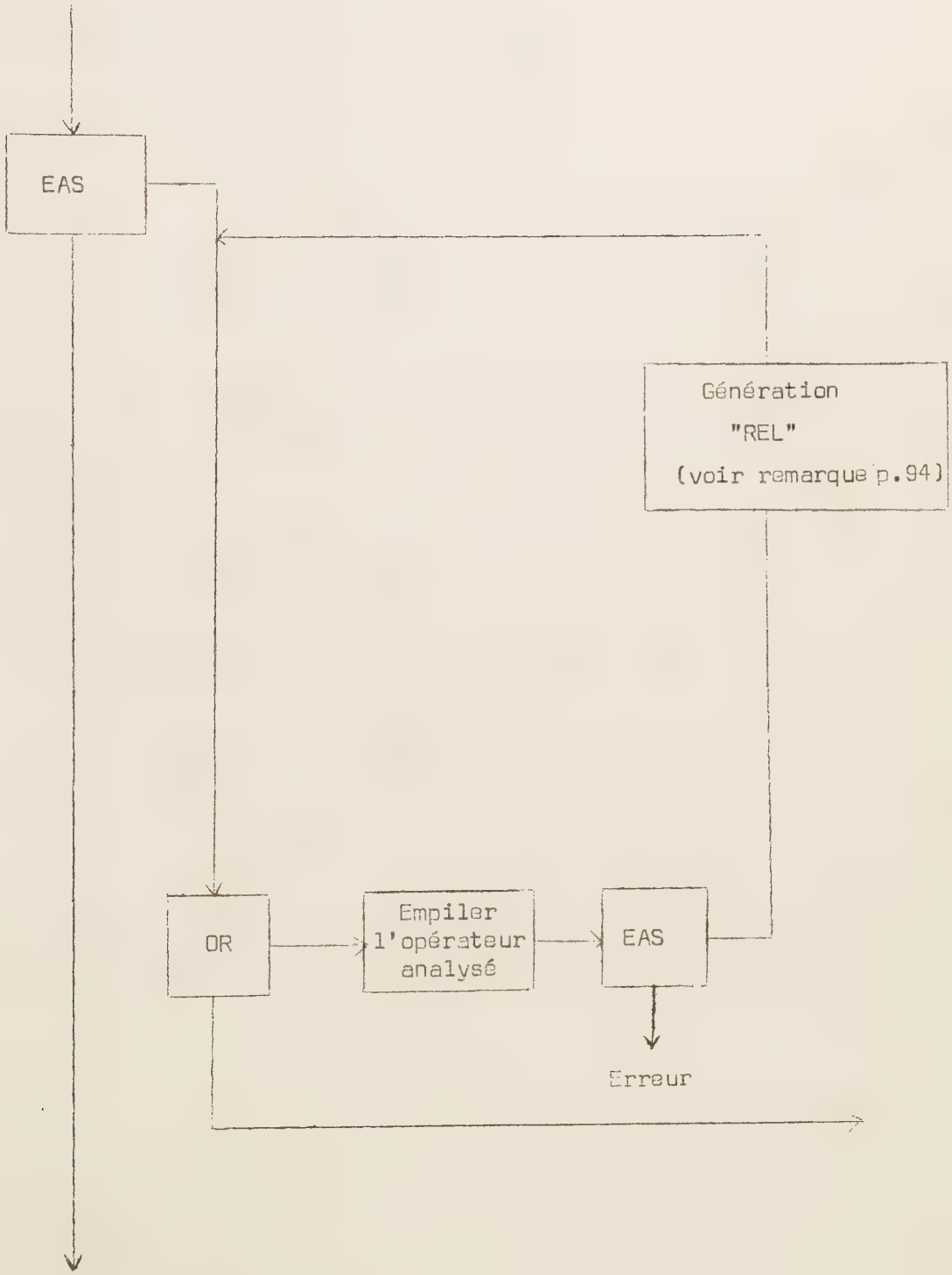
PRIMAIRE BOOLEEN



Nota :

La traduction d'une variable indicée sera analysée en même temps que celle arithmétique

RELATION


EAS
Génération
"REL"
(voir remarque p.94)
OR
Empiler
l'opérateur
analysé
EAS
Erreur

Traduction d'une expression booléenne conditionnelle

Soient B_1, B_2 , B_3 des expressions booléennes.

On a pour forme générale d'une expression booléenne conditionnelle :

"Si" B_1 "ALORS" B_2 "SINON" B_3

signifiant :

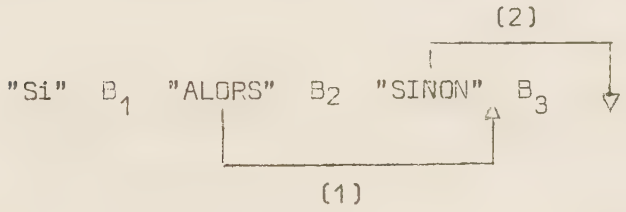
si B_1 est vrai, calculer B_2 ,

si B_1 est faux, calculer B_3

Le programme résultant étant construit sous forme linéaire ou sérielle, un saut inconditionnel du calcul de B_3 doit être prévu après le calcul de B_2 .

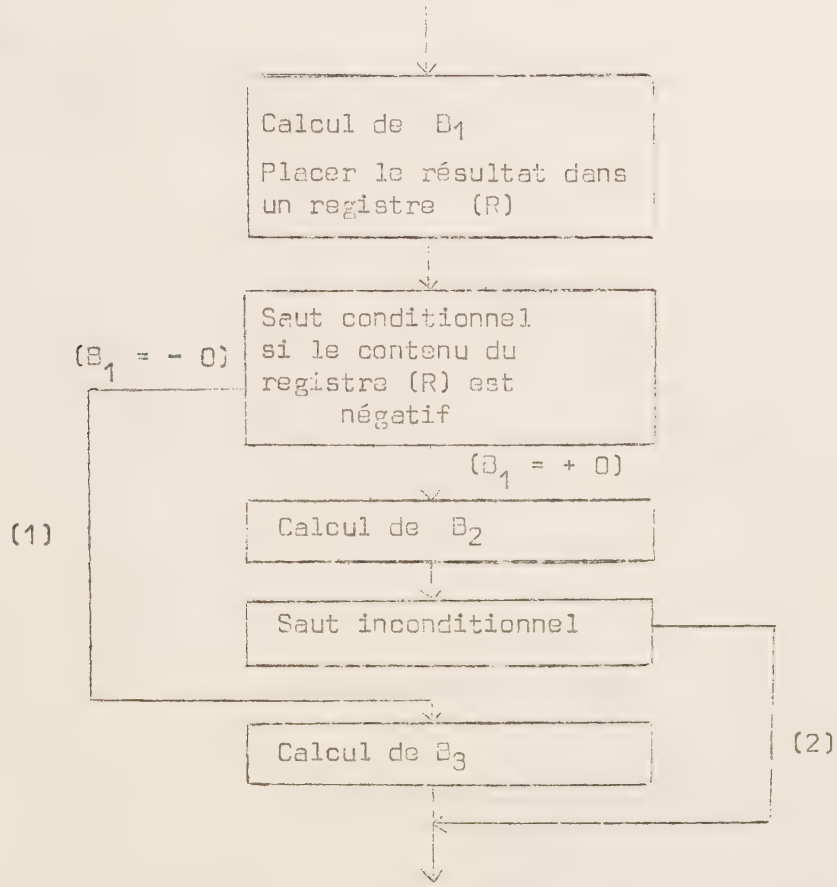
L'aiguillage sur le calcul de B_2 ou de B_3 , est fait par une instruction de saut conditionnel.

On peut schématiser le fonctionnement comme suit :



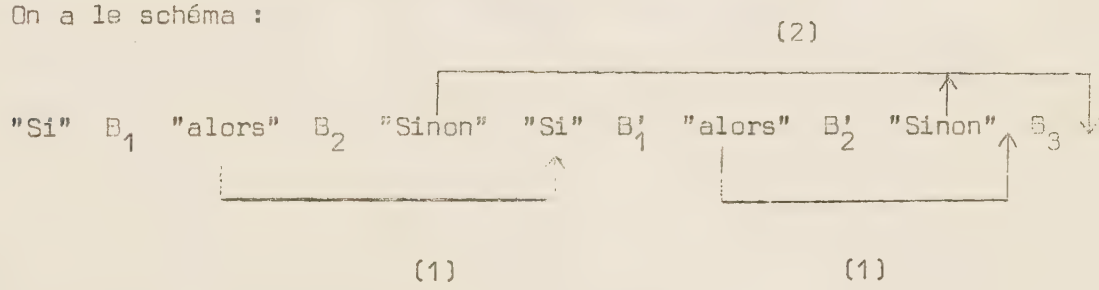
Le saut (1) conditionnel peut être considéré comme la traduction d'"ALORS", le saut (2) inconditionnel comme la traduction de "SINON"

On a alors l'organigramme du programme à obtenir



Imbrications des expressions booléennes conditionnelles

On a le schéma :



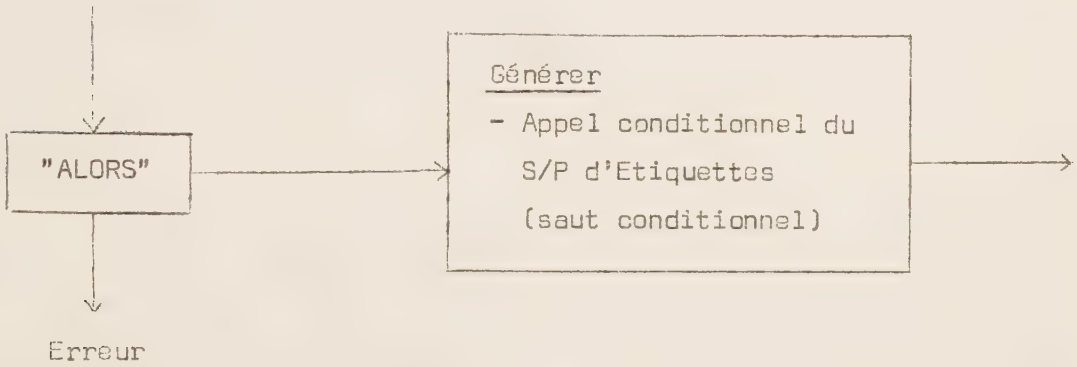
Les sauts (1) sont traités par une pile.

Le saut (2) est repéré seulement à la fin de l'expression complète.

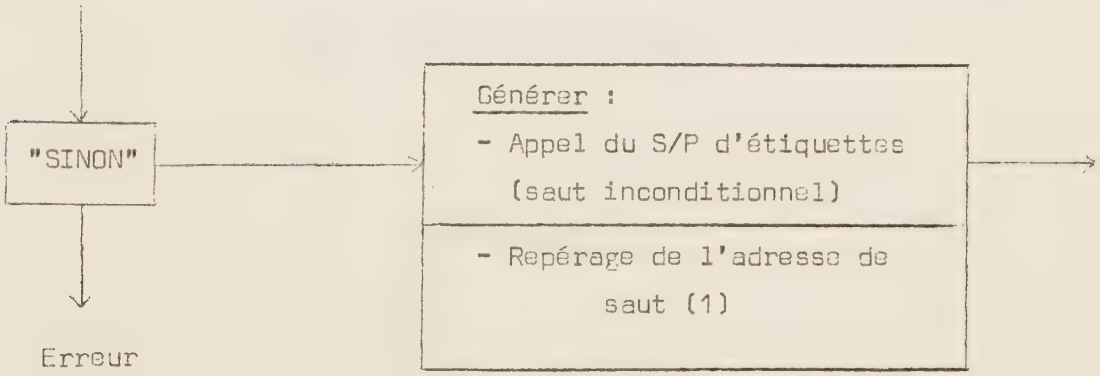
Traduction d'"ALORS"

Nota :

On utilise un sous-programme pour les sauts vers étiquettes à cause de la constitution de la mémoire PB 250.



Traduction de "SINON"



EXPRESSION DE DESIGNATION

Sémantique

Une expression de désignation est une règle pour obtenir l'étiquette d'une instruction.

Le principe d'évaluation est tout à fait analogue à celui qui concerne les expressions arithmétiques.

Traduction

N'ayant pas traité les aiguillages pourl'instant, nous ne verrons que le traitement des étiquettes.

La traduction d'une étiquette d'une expression de désignation est tout simplement une adresse de programme, qui se trouve dans la table des étiquettes.

TRADUCTION DES INSTRUCTIONS

Seules sont à traduire, les instructions suivantes :

- Instructions conditionnelles
- Instructions "pour"
- Instructions "Aller à"
- Instructions d'affectation
- Instructions procédures

INSTRUCTIONS CONDITIONNELLES

Sémantique

Il y a deux formes possibles d'instructions conditionnelles.

Soient B : une expression booléenne
II : une instruction inconditionnelle
IP : une instruction pour
I : une instruction quelconque

On a les deux formes :

1 - "Si" B "ALORS" II ;
ou "Si" B "ALORS" IP ; (de la même forme que ci-dessus)

2 - "Si" B "ALORS" II "SINON" I ;

sous forme schématique, on peut tracer le fonctionnement :

1 - "Si" B "ALORS" II ;

2 - "Si" B "ALORS" II "SINON" I ;

La traduction revient alors la même que celle des expressions conditionnelles, sauf dans le cas (1), où "SINON" n'intervient pas autrement, seuls les sauts "alors" et "sinon" sont à traduire.

INSTRUCTIONS POUR

Sémantique

Les éléments d'une liste de pour donnent trois types de fonctionnement.

1 - Type, E A :

Soient : B, C, ..., N une suite d'expressions arithmétiques

Une instruction pour, du premier type, a la forme :

```
"POUR" V := B, C, ..., N "FAIRE" Instruction S ;
```

qui signifie :

```
V  := B ; Instruction  S ;
V  := C ; Instruction  S ;
........
V  := N ; Instruction  S ;
```

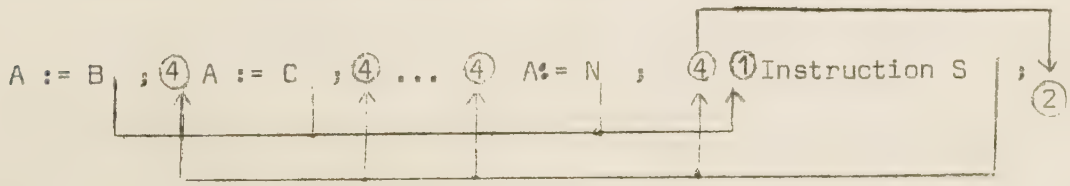
Ceci se simplifie en utilisant des transferts inconditionnels et un aiguillage.

Soit Aig [i] :=1, 2, ..., n un aiguillage.

On peut écrire l'instruction pour sous la forme :

```
i:= 1 ; V:= B ; "ALLER A" IS ;
i:= 2 ; 1 : V:= C ; "ALLER A" IS ;
.........
i:= n ; n-1 : V:= N ; "ALLER A" IS ;
        n   : "ALLER A" Instruction suivante ;
IS : Instruction S ; "ALLER A" Aig[ i ] ;
Instruction suivante : suite du programme
```

sous forme schématique, on peut tracer le fonctionnement :



2 - Type, EA "PAS" EA "JUSQU'A" EA

Toujours en reprenant la sémantique du rapport ALGOL 60, soient : A, B , C des expressions arithmétiques.

Une instruction pour, du deuxième type, a la forme :

```
"POUR" V := A "PAS" B "JUSQU'A" C "FAIRE" Instruction  S ;
```

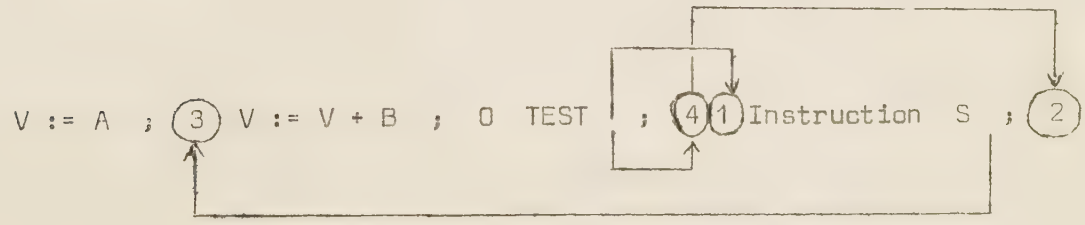
qui signifie :

```
V := A
L1 : "Si" (V-C) X signe (B) > 0 "ALORS" "ALLER A" Instruction
                                                   suivante ;
Instruction  S ;
V := V + B ;
"ALLER A" L1 ;
Instruction suivante : suite du programme
```

Ceci est transformé, pour des raisons bien simples, de traduction séquentielle, sous la forme :

```
V := A
"ALLER A" L2 ;
L1 : V := V + B ;
L2 : "Si" (V - C) X signe (B) > 0 "ALORS" "ALLER A" Instruction
                                                   suivante ;
Instruction S ;
"ALLER A" L1 ;
Instruction suivante : suite du programme.
```

Sous forme schématique, on a :



3 - Type , EA "TANT QUE" EB

Soient E : expression arithmétique

F : expression booléenne

Une instruction pour, du troisième type, a la forme :

"POUR" V := E "TANT QUE" F "FAIRE" Instruction S ;

qui signifie :

L3 : V := E ;

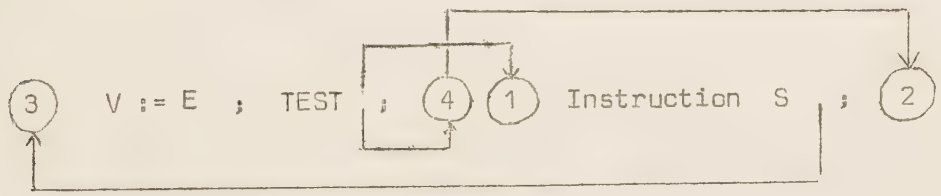
"Si" ¬ F "ALORS" "ALLER A" Elément épuisé ;

Instruction S ;

"ALLER A" L3 ;

Elément épuisé : suite du programme

sous forme schématique, on a :



Traduction des instructions "POUR"

On utilise des sous-programmes à l'exploitation dans les boucles POUR.

Ces sous-programmes étant les mêmes pour les différentes boucles, celles-ci pouvant être imbriquées, on crée des piles à l'exploitation.

Ces piles servent pour stocker :

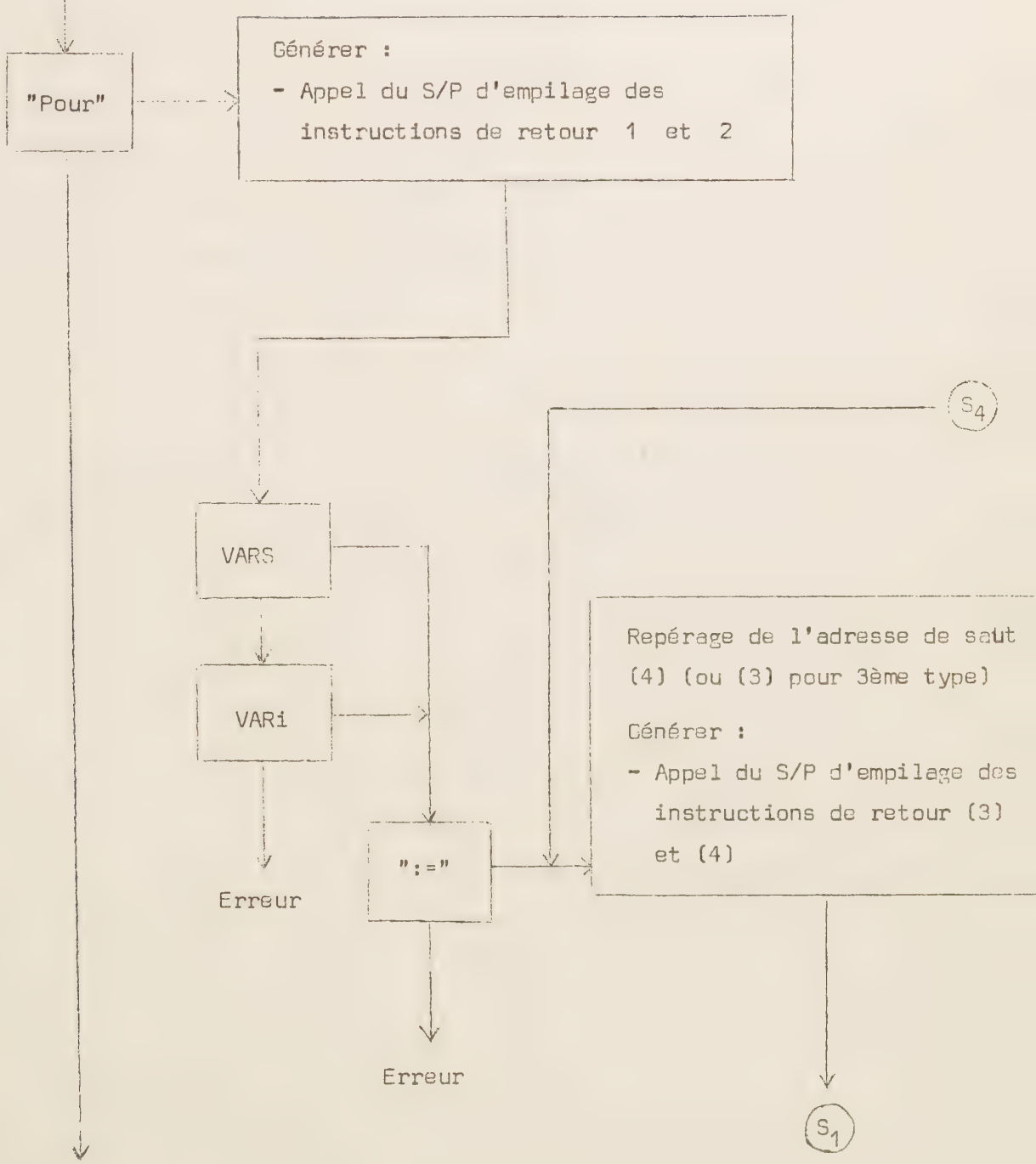
1) l'adresse de début de l'instruction située après "Faire"
2) l'adresse de fin de cette instruction
3) l'adresse de retour à l'élément de liste de pour en cours (pour les types 2 et 3)
4) l'adresse de retour à l'élément de liste suivant.

Ces adresses sont empilées à l'exploitation par bloc de deux (1 et 2, 3 et 4), par des sous-programmes.

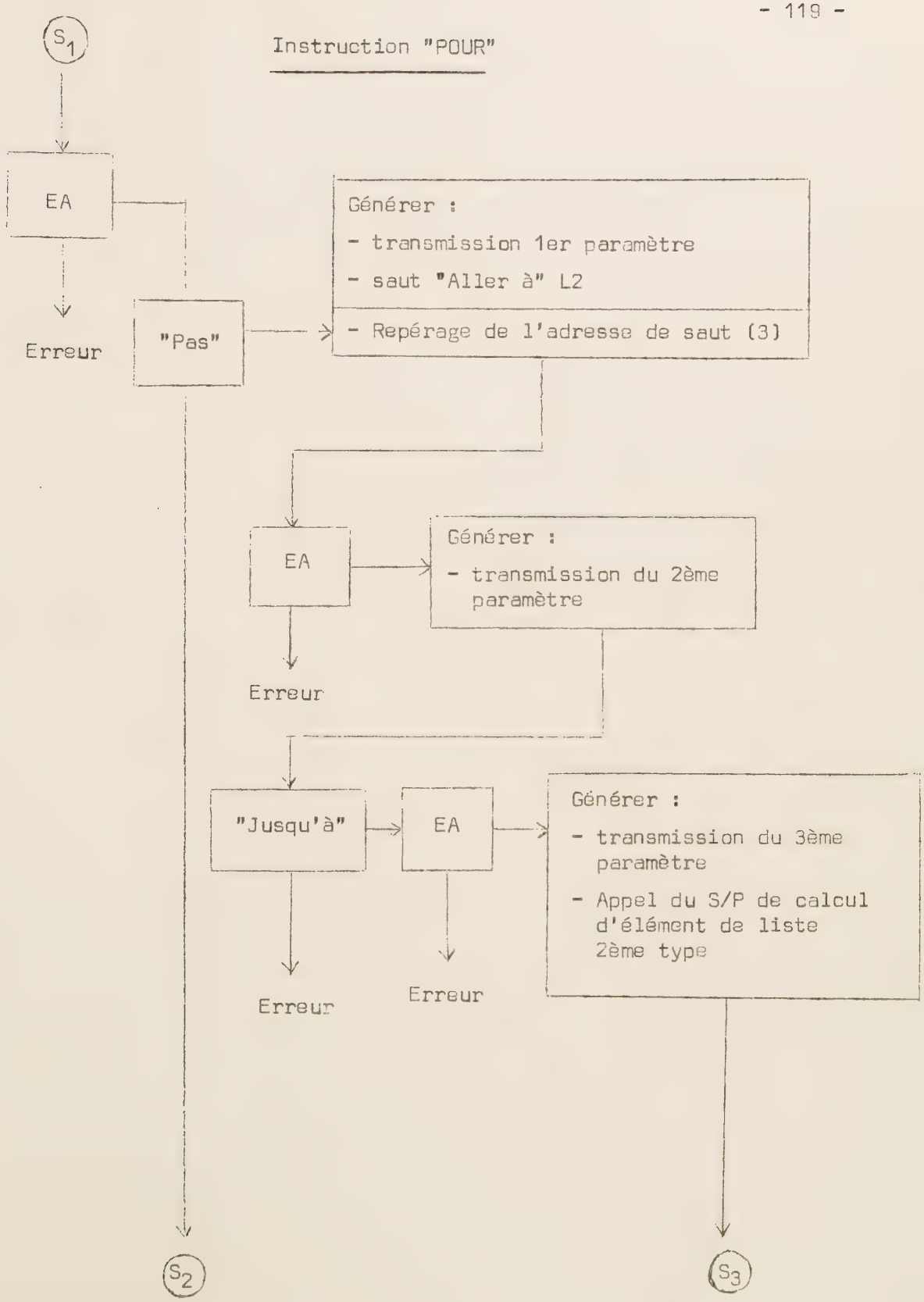
Les éléments de listes de pour sont déterminés aussi par sous-programmes à l'exploitation.

On obtient alors l'organigramme de génération :

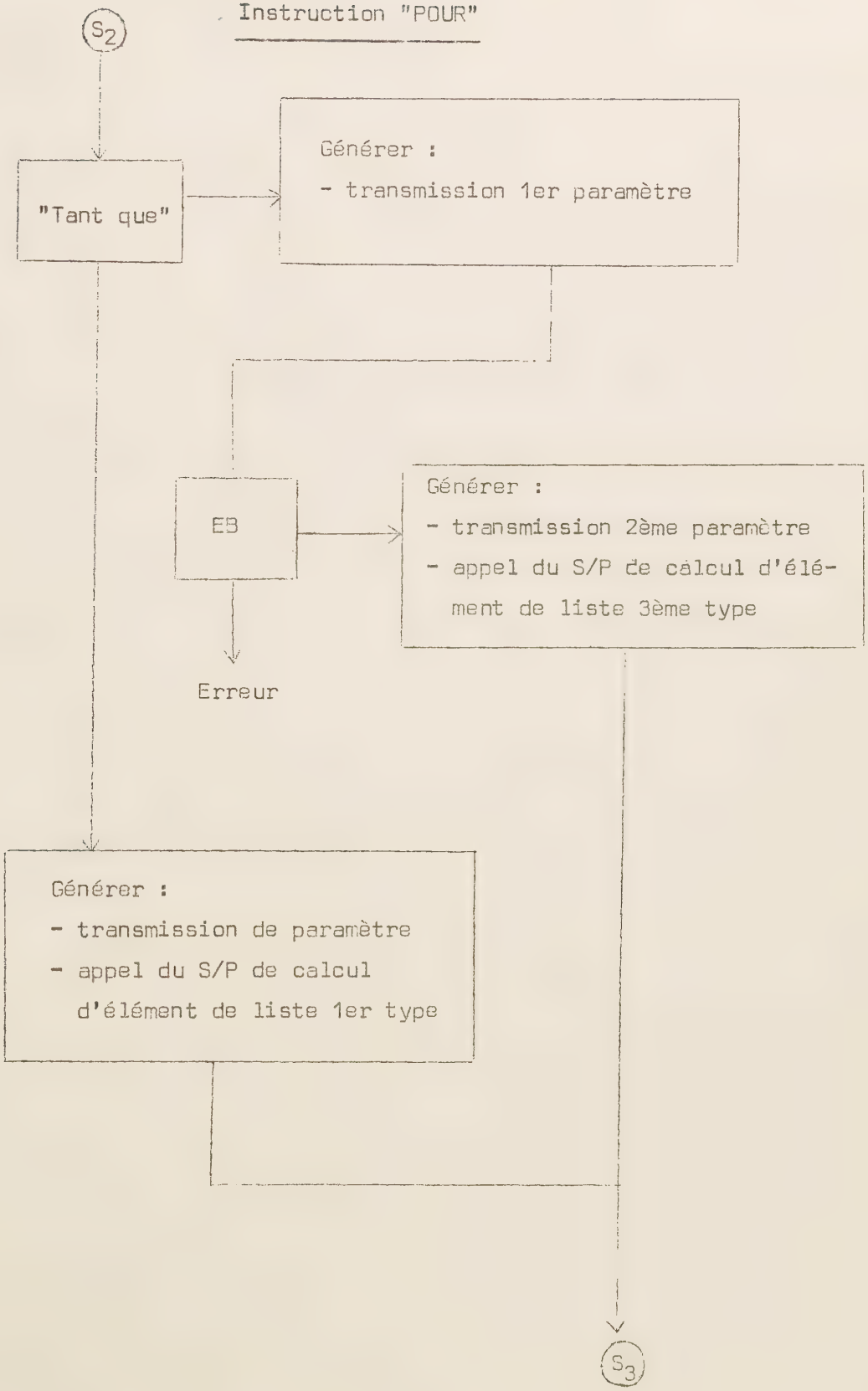
Instruction "POUR"



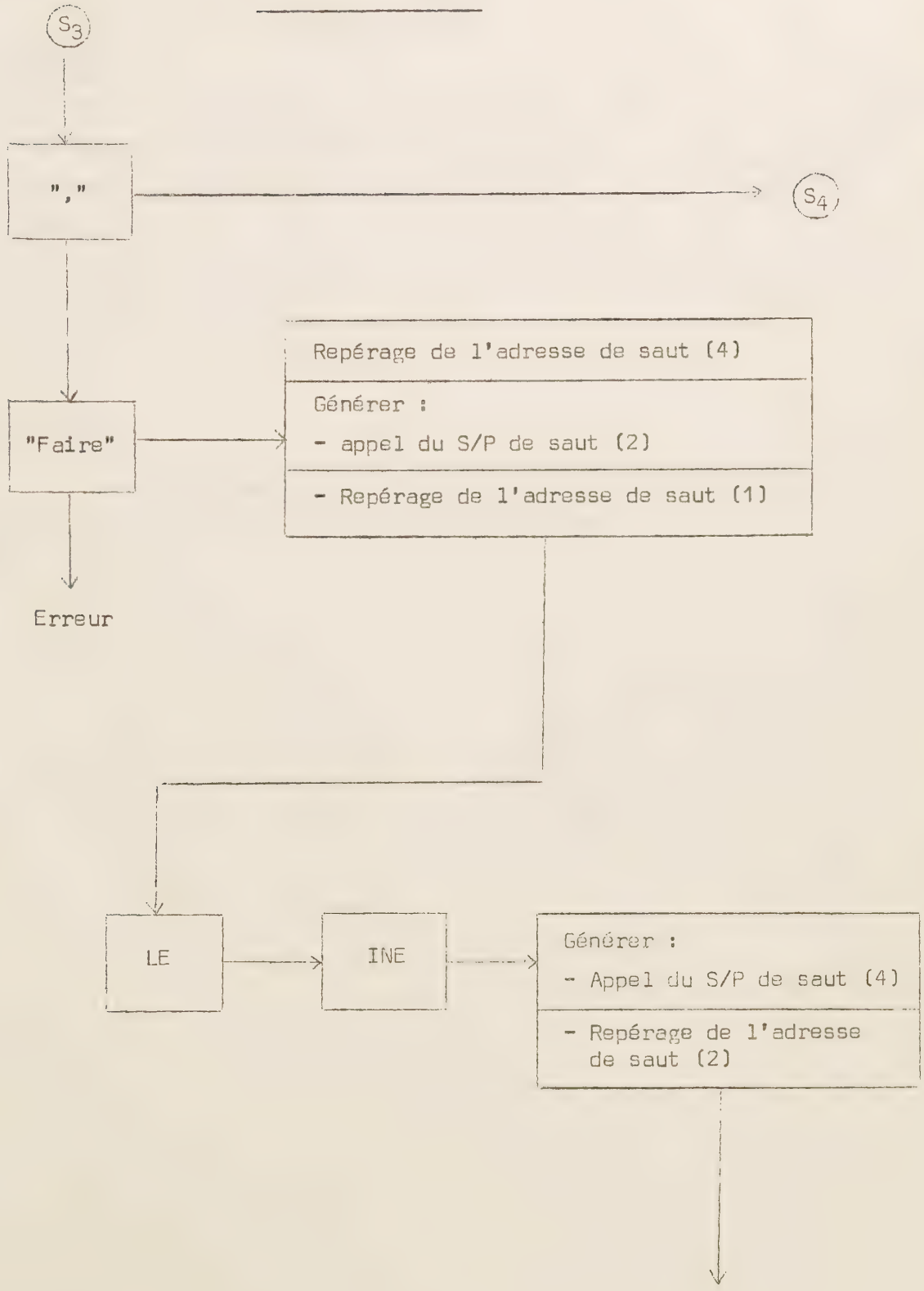
Instruction "POUR"



Instruction "POUR"



Instruction "POUR"



Instruction "ALLER A"

Sémantique

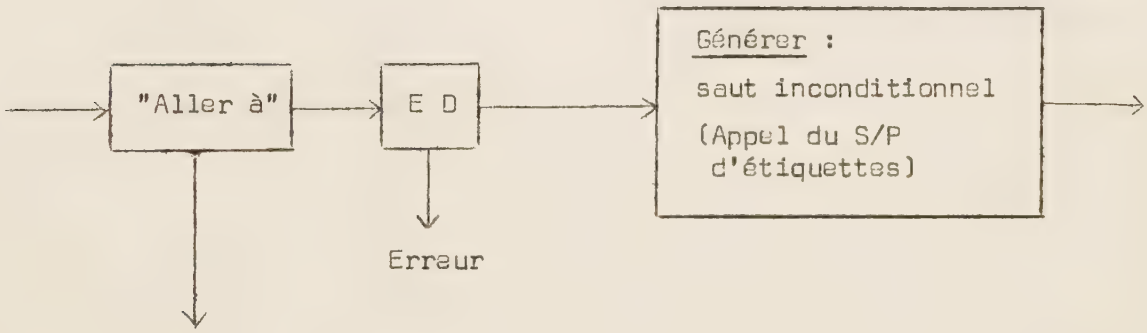
Une instruction "Aller à" interrompt la suite normale des opérations définie par l'ordre d'écriture des instructions, en définissant son successeur explicitement au moyen de la valeur d'une expression de désignation.

L'instruction suivante à exécuter sera celle ayant pour étiquette la valeur de cette expression.

Traduction

L'équivalent sémantique en langage machine est tout simplement un saut inconditionnel vers l'instruction suivante à exécuter

On a :



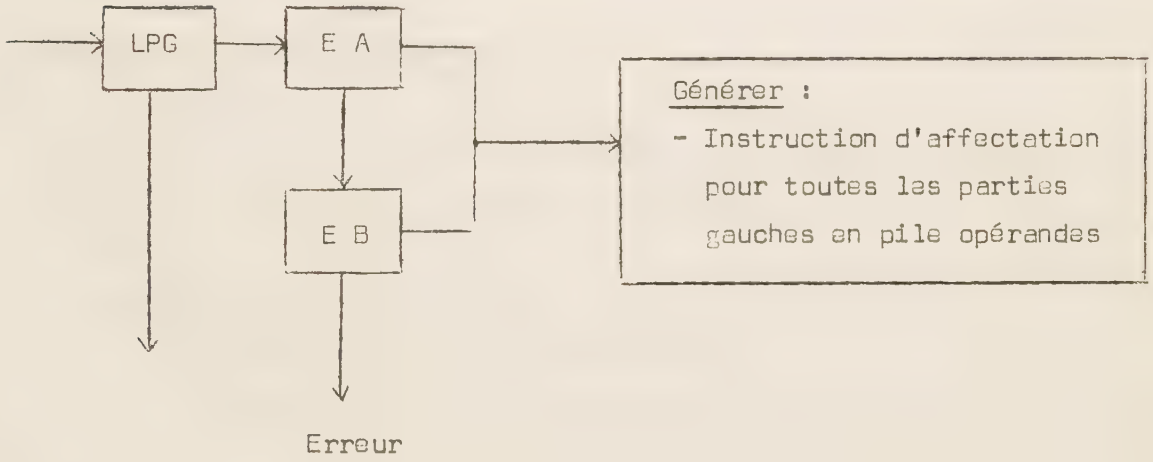
INSTRUCTION D'AFFECTATION

Traduction

Pour chaque partie gauche, une instruction machine d'affectation de valeur est générée, après traduction de la partie droite.

Ces parties gauches sont stockées en pile opérandes et reprises dans l'ordre inverse.

On a l'organigramme :



DECLARATIONS

Portée des identificateurs

Chaque bloc est numéroté dans l'ordre croissant.

Chaque identificateur est constitué de son code lu auquel on adjoint le numéro de bloc où il est déclaré.

A ce moment, la comparaison des identificateurs est immédiate.

Si n est le numéro de bloc associé à un identificateur d'une expression, il faut qu'il existe un identificateur déclaré avec un numéro de bloc associé n' tel que :

$$n' \leqslant n$$

Traitement des identificateurs

en déclaration

On construit leurs codes machine, on vérifie la double déclaration et on les stocke simplement en mémoire.

dans les expressions

On construit leurs codes machine

On vérifie s'ils ont été déclarés

TRAITEMENT DES TABLEAUX

Les tableaux pouvant être à bornes variables, c'est à l'exploitation que sont calculés l'encombrement et l'adresse initiale.

Ceci est effectué toujours par sous-programme de même que le calcul de l'adresse d'un élément de tableau.

Tous les paramètres intervenant dans ces calculs sont placés sous forme de vecteurs dans des tables (de bornes et d'indices)

On rappelle ci dessous les calculs afférant aux tableaux à l'exploitation.

Calcul de l'encombrement et de l'adresse initiale d'un tableau

Soient : $a_1 \quad a_2 \quad \ldots \quad a_n$: bornes inférieures

$b_1 \quad b_2 \quad \ldots \quad b_n$: bornes supérieures

avec $b_i \geqslant a_i$

Encombrement d'un tableau

On a :

$$\boxed{E = \prod_{i=1}^{n} \{b_i - a_i + 1\}}$$

Adresses initiales

Soient $T_1 \quad T_2 \quad \ldots \quad T_p$: tableaux appartenant à un même bloc

$E_1 \quad E_2 \quad \ldots \quad E_p$: encombrements correspondants

A partir d'une origine A_1 , on détermine les adresses initiales de chaque tableau par :

T_1 : adresse initiale A_1

T_2 : adresse initiale $A_2 = (A_1 - E_1)$

.....

T_i : adresse initiale $A_i = (A_{i-1} - E_{i-1})$

Calcul de l'adresse d'un élément de tableaux

On utilise un schéma de Horner :

Soient : $a_1 \quad a_2 \quad \ldots \quad a_n$: bornes inférieures

$b_1 \quad b_2 \quad \ldots \quad b_n$: bornes supérieures

$q_1 \quad q_2 \quad \ldots \quad q_n$: indices

A_o : Adresse initiale du tableau

En adoptant pour le tableau un mode de rangement linéaire colonne par colonne (avec un sens décroissant) à partir d'une adresse de référence A_o , l'adresse d'un élément courant $A_{q_1\, q_2 \ldots q_n}$, s'obtient par un schéma de Hörner :

On a :

$$\boxed{A_{q_1\, q_2 \ldots q_n} = A_o - \Delta A}$$

avec :

$$\Delta A = (q_n - a_n) \cdot (b_{n-1} - a_{n-1} + 1) \cdot (b_{n-2} - a_{n-2} + 1) \ldots (b_1 - a_1 + 1)$$

$$+ (q_{n-1} - q_{n-1}) \cdot (b_{n-2} - a_{n-2} + 1) \cdot (b_{n-3} - a_{n-3} + 1) \ldots (b_1 - a_1 +$$

$$+ \ldots$$

$$+ (q_1 - a_1)$$

soit :

$$\Delta A = \sum_{i=n}^{1} \left\{ (q_i - a_i) \cdot \prod_{j=i-1}^{1} (b_j - a_j + 1) \right\}$$

en posant :

$$\delta_j = b_j - a_j + 1$$

$$\alpha_i = q_i - a_i$$

On a : $\Delta A = \sum_{i=n}^{1} \left\{ \alpha_i \cdot \prod_{j=i-1}^{1} \delta_j \right\}$

d'où le schéma de Hörner :

$\Delta A_n = \alpha_n$
$\Delta A_i = \Delta A_{i+1} \cdot \delta_i + \alpha_i$
$\Delta A_1 = \Delta A_2 \cdot \delta_1 + \alpha_1$

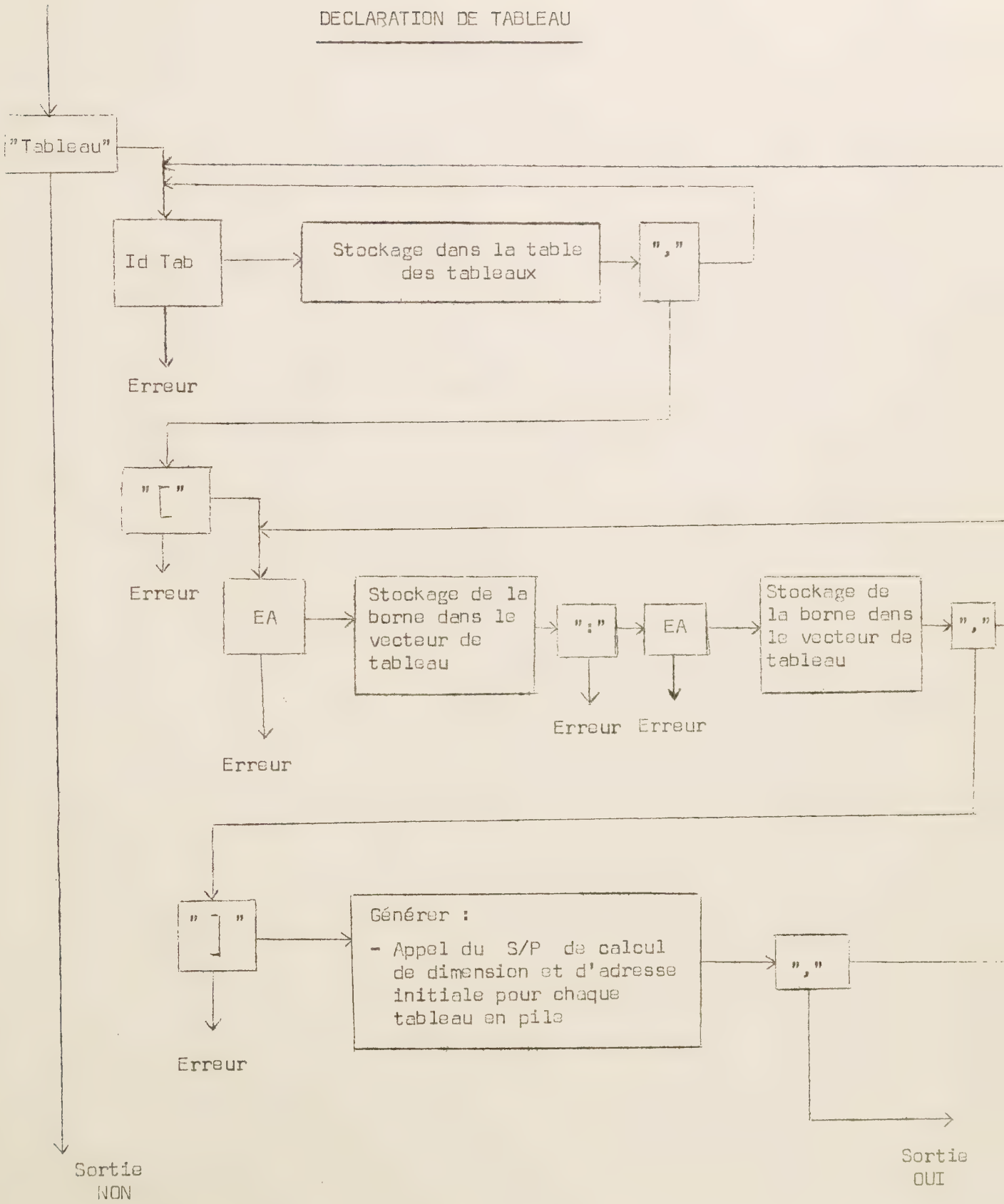
L'appel d'une variable indicée exige donc la connaissance :

- de l'adresse de référence A_o
- des bornes (a_i, b_i) , donc des dimensions
- des indices courants (q_i)

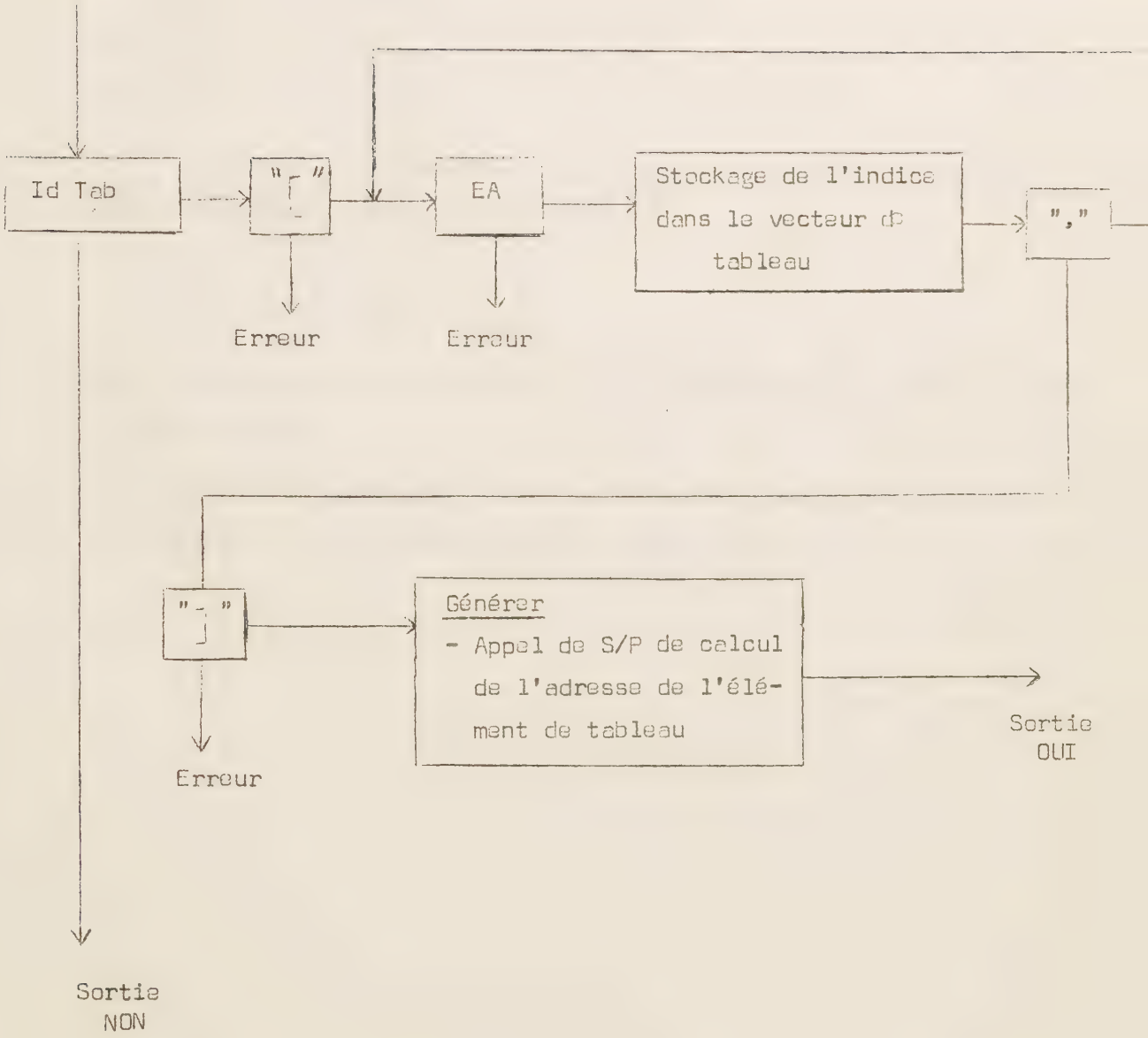
Traduction

Tous les renseignements sont stockés dans un vecteur mémoire, seuls l'origine et la longueur de ce vecteur interviennent. Le calcul se fait par appel de sous-programme.

DECLARATION DE TABLEAU



VARIABLE INDICEE



TRAITEMENT DES PROCEDURES

Seules des procédures simples ayant été traitées dans l'étude du compilateur, nous ne nous étenderons pas dans la description de la génération de celles-ci.

Les procédures traduites par le compilateur PB 250, sont tout de même utilisables. Elles permettent :

- un appel par valeur
- un appel par Nom

mais l'évaluation des paramètres ne se fait qu'une seule fois à l'entrée de la procédure.

Nous avons seulement à titre expérimental étudié les procédures récursives qui sont traitées par la méthode dite de Galler.

CONCLUSION

Le traducteur d'ALGOL que nous avons réalisé sur PB 250 n'est pas complet.

Les programmes traduits ont des limitations dues à la capacité de la mémoire, celle-ci n'est que d'environ 8 000 mots utilisables, dont seulement 3 000 mots de mémoire active. De plus, les restrictions du langage proviennent essentiellement de ce que le PB 250 n'a pas l'adressage indirect, utile pour traiter les tableaux, les aiguillages et les procédures.

Néanmoins, notre but était de construire un traducteur en vue de l'enseignement du langage ALGOL. Les exemples donnés en annexe montrent que ce traducteur est utilisable.

C'est surtout la simplification des équations syntaxiques qui nous a permis de construire ce compilateur. Sans cela, dans 3 000 mots de mémoire, il était à notre avis presque impossible de le réaliser.

Mais l'intérêt de la méthode est de pouvoir utiliser au maximum le calculateur dans la construction d'un traducteur.

Si l'on considère le problème de la reconnaissance d'un langage (L) , celle-ci est effectuée par un automate associé à un système d'équations syntaxiques (S) permettant d'obtenir (L) .

On peut envisager la construction de cet automate par le calculateur lui-même.

Le problème revient à construire un autre automate capable d'analyser le système (S) (exemple : ALGOL) et d'engendrer l'automate qui reconnaitra les mots (W $\in$ L).

La construction d'un analyseur syntaxique demandera alors quelques minutes de calculateur PB 250.

Un système d'équations syntaxiques (S) décrivant un langage (L) peut être considéré comme un mot (w') appartenant au monoïde libre V^* relatif à un système de générateurs libres

$$V = V_T \cup V_A$$

Où V_T : système de générateurs libres du langage L

V_A : système de générateurs libres auxiliaire

V_A est l'ensemble des variables du système d'équations (S) décrivant (L) et des symboles + , . , = , (et) intervenant dans l'écriture d'une équation.

Exemple : ALGOL

(1) EA = EAS + "Si" . EB . "Alors" . EAS . "Sinon" . EA

EA , = , EAS , + , . , EB $\in V_A$

"Si" , "Alors" , "Sinon" $\in V_T$

Nota :

Pour éviter des confusions, tous les générateurs libres du langage L sont écrits entre guillemets, et les symboles $\in$, + , . , = , (, et) sont entourés.

Le mot (W') appartient à un langage (L') qui peut être obtenu à partir du système d'équations (S') dans l'algèbre de $\mathcal{P}(V^*)$ suivant :

en posant :

SE : système d'équations syntaxiques
E : équation syntaxique
VL : variable libre $V_L \in V_A$
CL : constante libre $C_L \in V_T$
EXL : expression libre
TL : terme libre
PL : primaire libre
FS : fonction standard : "Opération Etoile"

En introduisant un symbole séparateur d'équations

" $ " $\in V_A$

On a :

(1) $SE = E^+ = E \,.\, E^*$

(2) $E = VL \,.\, (=) \,.\, EXL \,.\, (\$)$

(3) $EXL = TL \,.\, ((+) \,.\, TL)^*$

(4) $TL = PL \,.\, ((.) \,.\, PL)^*$

(5) $PL = VL + CL + FS + ((\,.\, EXL \,.\, (\,)$

Traduit sous forme de Backus, on obtient la syntaxe :

< Système d'équations > ::= < Système d'équations > < Equation > |
< Equation >

< Equation > ::= < Variable libre > (=) < Expression libre > ($)

< Expression libre > ::= < Expression libre > (+) < Terme libre > |
< Terme libre >

< Terme libre > ::= < Terme libre > (.) < Primaire libre > | < Primaire libre >

< Primaire libre > ::= < Variable libre > | < Constante libre > |
< Fonction standard > | < Expression libre >

< Constante libre > := symbole appartenant à V_T

< Variable libre > ::= symbole appartenant à V_A

Nous ne développerons pas ici l'étude d'un tel compilateur qui serait analogue à celle du compilateur ALGOL étudié .

Nous envisageons de la poursuivre en vue de construire un analyseur syntaxique d'ALGOL sur PB 250 , mais aussi de construire des analyseurs pour d'autres types de calculateurs.

BIBLIOGRAPHIE

1 - J. KUNTZMANN — Théorie des relations et des réseaux
Cours de l'IMAG - Grenoble - 1966

2 - A.G. KUROSH — Algèbre générale (traduit par J.P. Peaudecerf)
Dunod - 1967

3) J. CHINAL — Techniques Booléennes et calculateurs arithmétiques
Dunod - 1967

4) M. CROSS et A. LENTIN - Notions sur les grammaires formelles
Gauthier-Villars - 1967

5) B. VAUQUOIS — Logique et Programmation
Cours de l'IMAG - Grenoble - 1964

6) L. BOLLIET, N. GASTINEL, PJ. LAURENT
Un nouveau langage scientifique : ALGOL
Manuel pratique
Hermann - 1964

7) L. BOLLIET — L'écriture des compilateurs
Revue Française du traitement de l'information :
Chiffres - N° 1 et 2 - 1966

8) M.P. SCHUTZENBERGER - Automates et langages formels
Revue du CETHEDEC - N° 1 - 1964

9) P. NAMIAN — Machines mathématiques
Cours du C.N.A.M. - Paris - 1967

A N N E X E

EXEMPLES DE PROGRAMMES

D'ANALYSE NUMERIQUE

UTILISABLES SUR PB 250

```
'DEBUT'
'COMMENTAIRE':INTERPOLATION PAR LES METHODES DE NEWTON-GREGORY ET DE BESSEL;
'ENTIER'I,J,N,M;
'REEL'X0,H,BI,X,CI,EI,U,P,UC,F,EM,Q;
LIRE(N);
'DEBUT''TABLEAU'B,Y[0:N+2],D[0:N,0:N];
LIRE(X0,H,BI);
'POUR'I=0'PAS'1'JUSQUA'N'FAIRE'
'DEBUT'LIRE(Y[I]);D[0,I]=Y[I]'FIN';
'POUR'J=1'PAS'1'JUSQUA'N'FAIRE'
'DEBUT'X=2πJ*BI;
IMPRIMER('OUV'DIFFERENCES D ORDRE'FER',J,'OUV'AVEC INCERTITUDE'FER',X);
'POUR'I=0'PAS'1'JUSQUA'N-J'FAIRE'
'DEBUT'D[J,I]=D[J-1,I+1]-D[J-1,I];IMPRIMER(D[J,I])'FIN' 'FIN';
'COMMENTAIRE' FIN DU CALCUL DES DIFFERENCES;
UN:LIRE(X,M,Q,CI);
EI=CI*BI;
'SI'Q'EGAL'1'ALORS''ALLERA' BESSEL;
U=UC=(X-X0)/H;
P=Y[0]+U*D[1,0];
F=J=1;
DEUX:J=J+1;
F=F*J;
UC=(U-J+1)*UC;
```

```
'SI'J'EGAL'M+1 'ALORS' 'ALLERA' TROIS;
P=P+(UC/F)*D[J,0];
'ALLERA' DEUX;
TROIS:EM=ABS(UC/F)*(ABS(D[J,0])+2πJ*BI);
IMPRIMER(X,P,EM,EI);
'ALLERA' UN;
'COMMENTAIRE' FIN DE LA METHODE DE NEWTON-GREGORY;
BESSEL:P=(Y[(N-1)/2]+Y[(N-1)/2+1])/2;
U=(X-(X0+H*(N-1)/2)-H/2)/H;
B[0]=1;B[1]=U;
'POUR'I=2' PAS' 2 'JUSQUA'
('SI' M/2 'EGAL' ENTIER(M/2) 'ALORS' M+2 'SINON' M+1) 'FAIRE'
'DEBUT' B[I]=B[I-2]*(Uπ2-((I-1)/2)π2)/(I*(I-1));
B[I+1]=B[I]*U/(I+1) 'FIN';
'POUR' I=1 'PAS' 1 'JUSQUA' M 'FAIRE'
P=P+B[I]*('SI' I/2 'EGAL'ENTIER( I / 2 )'ALORS'
 (D[I,(N-I-1)/2]+D[I,(N-I-1)/2+1])/2
'SINON' D[I,(N-I)/2]);
IMPRIMER(P,EI,B[M+1],B[M+2]);
'ALLERA' UN;
'COMMENTAIRE' FIN DE LA METHODE DE BESSEL;

'FIN''FIN'
```

```
'DEBUT''COMMENTAIRE'RESOLUTION D UNE EQUATION PAR LA METHODE DE BAIRSTOW;
'REEL'DELTA,PR,PI,D,DS,DP,Q;
'PROCEDURE'TRINOME(S,P,X1,Y1,X2,Y2);'REEL'S,P,X1,Y1,X2,Y2;
'DEBUT'DELTA=S*S-4*P;PR=S/2;PI=RAC(ABS(DELTA))/2;
'SI'DELTA'SPG'0'ALORS''DEBUT'X1=PR+PI;X2=PR-PI;Y1=Y2=0'FIN'
'SINON''DEBUT'X1=X2=PR;Y1=PI;Y2=-PI'FIN''FIN';
'ENTIER'N,J,R;
'REEL'SO,PO,EPSILON,SS,PP,XX1,YY1,XX2,YY2;
UN:  LIRE(N);
'DEBUT''TABLEAU'A[0:N],B,C[-1:N];
'POUR'J=0'PAS'1'JUSQUA'N'FAIRE'LIRE(A[J]);
DEUX:  LIRE(SO,PO,EPSILON);
PP=PO;SS=SO;R=0;
B[-1]=0;C[-1]=0;B[0]=A[0];C[0]=A[0];
TROIS:  'POUR'J=1'PAS'1'JUSQUA'N-1'FAIRE'
'DEBUT'B[J]=A[J]+SS*B[J-1]-PP*B[J-2];
C[J]=B[J]+SS*C[J-1]-PP*C[J-2]'FIN';
B[N]=A[N]+SS*B[N-1]-PP*B[N-2];
D=C[N-2]↑2+C[N-3]*(B[N-1]-C[N-1]);
'SI'D'EGAL'0'ALORS''DEBUT'IMPRIMER(D);'ALLER A'DEUX'FIN';
DS=(B[N]*C[N-3]-B[N-1]*C[N-2])/D;
DP=(B[N]*C[N-2]+B[N-1]*(B[N-1]-C[N-1]))/D;
SS=SS+DS;PP=PP+DP;
'SI'(ABS(DP)+ABS(DS))/(ABS(PP)+ABS(SS))'IFG'EPSILON'ALORS''ALLER A'QUATRE;
```

```
R=R+1;
'SI'R'INFERIEUR'50'ALORS''ALLER A'TROIS;
IMPRIMER('OUV'NON CONVERGENCE EN 50 ITERATIONS'FER,SS,PP);
LIRE(Q);
'SI'Q'EGAL'1'ALORS''ALLER A'UN'SINON''ALLER A'DEUX;
QUATRE:TRINOME(SS,PP,XX1,YY1,XX2,YY2);
IMPRIMER(R,XX1,YY1,XX2,YY2);
N=N-2;
'SI'N'IFG'2'ALORS''ALLER A'CINQ;
'POUR'J=1'PAS'1'JUSQUA'N'FAIRE'
A[J]=B[J];R=0;'ALLER A'TROIS;
CINQ:'SI'N'EGAL'2'ALORS''DEBUT'SS=-B[1]/B[0];PP=B[2]/B[0];
IMPRIMER(R);
TRINOME(SS,PP,XX1,YY1,XX2,YY2);
IMPRIMER(XX1,YY1,XX2,YY2);
'ALLER A'UN'FIN';
XX1=-B[1]/B[0];
YY1=0;
IMPRIMER(XX1,YY1);
'ALLER A'UN;
'FIN'
'FIN'
```

```
'DEBUT''COMMENTAIRE'RESOLUTION D UN SYSTEME PAR LA METHODE DE GAUSS;
'ENTIER'N;'REEL'EPS;  LIRE(N,EPS);
'DEBUT'  'TABLEAU' A[1:N+1,1:N],X[1:N+1];
'ENTIER'I,J,K;
'POUR'J=1'PAS'1'JUSQUA'N'FAIRE'
'POUR'I=1'PAS'1'JUSQUA'N+1'FAIRE'
LIRE(A[I,J]);
IMPRIMER('OUV'FIN DES DONNEES'FER');
'POUR'K=1'PAS'1'JUSQUA'N'FAIRE'
'DEBUT'J=K;ETI:'SI'J'EGAL'N+1'ALORS''ALLERA'IMPOSSIBLE;
'SI'ABS(A[K,J])'INFERIEUR'EPS'ALORS''DEBUT'J=J+1;'ALLERA'ETI'FIN';
'SI'J'DIFFERENT'K'ALORS''POUR'I=1'PAS'1'JUSQUA'N+1'FAIRE'
'DEBUT'X[I]=A[I,J];A[I,J]=A[I,K];A[I,K]=X[I]'FIN';
'POUR'J=K+1'PAS'1'JUSQUA'N+1'FAIRE'
'DEBUT'A[J,K]=A[J,K]/A[K,K];
'POUR'I=K+1'PAS'1'JUSQUA'N'FAIRE'
A[J,I]=A[J,I]-A[K,I]*A[J,K]'FIN''FIN';
'POUR'K=1'PAS'1'JUSQUA'N'FAIRE'
'DEBUT'J=N+1-K;
X[J]=A[N+1,J];
'SI'J'DIFFERENT'N'ALORS'
'POUR'I=J+1'PAS'1'JUSQUA'N'FAIRE'
X[J]=X[J]-A[I,J]*X[I];
IMPRIMER(J,X[J])'FIN';'ALLER A'FIN;
IMPOSSIBLE:  IMPRIMER('OUV'IMPOSSIBLE'FER');
FIN:  'FIN' 'FIN'
```

```
'DEBUT''COMMENTAIRE' METHODES DE RELAXATION;
'ENTIER' N;LIRE(N);
'DEBUT''REEL' C,Q,U, EPS,NH,DH, NR,B1,B2;'ENTIER'I,J,K;
'TABLEAU'X,R,M[1:N] , A[1:N,1:N+1];
IMPRIMER('OUV'COEFFICIENTS DU SYSTEME'FER');
'POUR'I=1 'PAS'1'JUSQUA'N'FAIRE''POUR'J=1'PAS'1'JUSQUA'N+1 'FAIRE'
'DEBUT' LIRE (A[I,J]); IMPRIMER (A[I,J]) 'FIN';
E1:LIRE (Q, B1, B2, U, EPS);
'POUR' I =1 'PAS' 1 'JUSQUA' N 'FAIRE' X[I]=U;C=0;
E2:C=C+1;'SI'C'SUP'Q 'ALORS''DEBUT'IMPRIMER('OUV'NON CONVERGENCE'FER');
'ALLERA' E1 ;'FIN';K=0;
E3:K=K+1;'POUR' I=1 'PAS' 1 'JUSQUA' N 'FAIRE'
'DEBUT' R[I]=A[I,N+1];
'POUR' J=1 'PAS' 1 'JUSQUA' N 'FAIRE' R[I]=A[I,J]*X[J]+R[I] 'FIN';
'SI' B2'EGAL' 1 'ALORS'
'DEBUT' M[K]=-R[K]/A[K,K];'ALLERA' E4 'FIN';NH=DH=0;
'POUR'I=1 'PAS' 1 'JUSQUA' N 'FAIRE''DEBUT' NH=R[I]*A[I,K]+NH;
DH=A[I,K]π2+DH 'FIN';M[K]=-NH/DH;
E4:X[K]=X[K]+M[K];'SI'K'INF'N 'ALORS' 'ALLERA' E3;NR=0;
'POUR' I=1 'PAS' 1 'JUSQUA' N 'FAIRE' NR=ABS(R[I])+NR;
'SI' B1'EGAL' 1 'ALORS' IMPRIMER (C,NR);
'SI' NR 'SUP' EPS 'ALORS' 'ALLERA' E2;IMPRIMER('OUV'RESULTATS'FER',C);
'POUR' I=1 'PAS' 1 'JUSQUA' N 'FAIRE' IMPRIMER (X[I]);
'ALLERA' E1;'FIN''FIN'
```

```
'DEBUT''COMMENTAIRE' INTEGRATION PAR LA METHODE DE NEWTON-COTES;
'ENTIER'K,N;'REEL'EM,EI,INT,A,B,Q,BI;
'TABLEAU'S[0:8,1:8];'TABLEAU'D,F[1:8];'TABLEAU'Y[0:8];
LIRE(Q);'POUR'N=1'PAS'1'JUSQUA'8'FAIRE'
'DEBUT'LIRE(D[N],F[N]);
'POUR'K=0'PAS'1'JUSQUA'ENTIER(N/2)'FAIRE'
'DEBUT'LIRE(S[K,N]);S[N-K,N]=S[K,N]'FIN';
'SI'Q'EGAL'1'ALORS''DEBUT'IMPRIMER(N,D[N],F[N]);
'POUR'K=0'PAS'1'JUSQUA'N'FAIRE'IMPRIMER(S[K,N])'FIN''FIN';
UN:LIRE(A,B,N,BI);
'POUR'K=0'PAS'1'JUSQUA'N'FAIRE''DEBUT'
IMPRIMER(K);LIRE(Y[K]);'FIN';
INT=0;'POUR'K=0'PAS'1'JUSQUA'N'FAIRE'
INT=INT+Y[K]*S[K,N];INT=(B-A)*INT/D[N];
EI=(B-A)*BI;
'SI'N/2'EGAL'ENTIER(N/2)'ALORS'EM=((B-A)/N)π(N+3)/F[N]
'SINON'EM=((B-A)/N)π(N+2)/F[N];
IMPRIMER(INT,EM,EI);
'ALLERA'UN;
'FIN'
```

```
'DEBUT''COMMENTAIRE' INTEGRATION PAR LA METHODE DE GAUSS;
'REEL''PROCEDURE'F(XI);'REEL'XI;
'COMMENTAIRE'L INSTRUCTION COMPRISE ENTRE LES DEUX POINTS VIRGULES QUI SUIVENT
(ET QUI PEUT ETRE SIMPLE OU COMPOSEE) DECRIT LA FONCTION A INTEGRER F(X) A
L AIDE DES PROCEDURES STANDARDS,ON A PRIS POUR EXEMPLE :
F(X)=LOG(TG(X/2)) QUI UTILISE LES PROCEDURES STANDARDS :
SIN,COS,ABS,LNE;
F=LNE(ABS(SIN(XI/2)/COS(XI/2)));
'TABLEAU'D,T[0:5];'TABLEAU'X,H[0:5,1:5];
'REEL'I,N;'REEL'A,B,S,CM;
'POUR'N=1'PAS'1'JUSQUA'5'FAIRE'
'DEBUT'IMPRIMER(N);LIRE(D[N]);
'POUR'I=0'PAS'1'JUSQUA''SI'(N/2)'EGAL'ENTIER(N/2)'ALORS'N/2'SINON'(N-1)/2
'FAIRE''DEBUT'LIRE(X[I,N],H[I,N]);X[N-I,N]=-X[I,N];
H[N-I,N]=H[I,N]'FIN';'POUR'I=0'PAS'1'JUSQUA'N'FAIRE'
'DEBUT'A=-X[I,N];IMPRIMER(A,H[I,N])'FIN''FIN';
E:LIRE(N,A,B);S=0;
'POUR'I=0'PAS'1'JUSQUA'N'FAIRE'
'DEBUT'T[I]=((B-A)*X[I,N]+B+A)/2;
S=S+H[I,N]*F(T[I])'FIN';
S=(B-A)*S/2;
CM=(B-A)π(2*N+3)/D[N];
IMPRIMER(S,CM);
'ALLERA'E'FIN'
```

```
'DEBUT'
'COMMMENTAIRE' INTEGRATION D EQUATIONS DIFFERENTIELLES PAR LA METHODE DE RUNGE-KUT
'ENTIER'N,Q;'REEL'T,TM,Y0,Z0,Y,Z,H,TE1,TE2,T1,T2,Y1,Y2,Z1,Z2,A,B,C,D;
'REEL''PROCEDURE'F(Y1,Z1,T1);'REEL'Y1,Z1,T1;
F=Z1;
'REEL''PROCEDURE'G(YJ,ZJ,TJ);'REEL'YJ,ZJ,TJ;
'DEBUT''SI'N'EGAL'1'ALORS'G=0'SINON'G=-3/2*ZJ-1/2*YJ;'FIN';
ORDRE:LIRE(N);'SI'N'EGAL'1'ALORS'Z0=1;
LIRE(T,H,TM,Y0);'SI'N'EGAL'2'ALORS'LIRE(Z0);LIRE(Q);
'SI'Q'EGAL'1'ALORS''ALLERA'RK1;
LIRE(TE1);'SI'Q'EGAL'2'ALORS''ALLERA'RK2;LIRE(TE2);
A=(3*TE2-2)/(6*TE1*(TE2-TE1));B=(2-3*TE1)/(6*TE2*(TE2-TE1));
C=TE2*(TE2-TE1)/(TE1*(2-3*TE1));D=1-A-B;'ALLER A'RK2;
RK1:Y=Y0+H*F(Y0,Z0,T);Z=Z0+H*G(Y0,Z0,T);'ALLER A'SORTIE;
RK2:T1=T+H*TE1;Y1=Y0+H*TE1*F(Y0,Z0,T);Z1=Z0+H*TE1*G(Y0,Z0,T);
'SI'Q'EGAL'3'ALORS''ALLER A'RK3;Y=Y0+H/(2*TE1)*((2*TE1-1)*F(Y0,Z0,T)+F(Y1,Z1,T1));
Z=Z0+H/(2*TE1)*((2*TE1-1)*G(Y0,Z0,T)+G(Y1,Z1,T1));'ALLER A'SORTIE;
RK3:T2=T+H*TE2;Y2=Y0+H*(TE2*F(Y0,Z0,T)+C*(F(Y1,Z1,T1)-F(Y0,Z0,T)));
Z2=Z0+H*(TE2*G(Y0,Z0,T)+C*(G(Y1,Z1,T1)-G(Y0,Z0,T)));
Y=Y0+H*(D*F(Y0,Z0,T)+A*F(Y1,Z1,T1)+B*F(Y2,Z2,T2));
Z=Z0+H*(D*G(Y0,Z0,T)+A*G(Y1,Z1,T1)+B*G(Y2,Z2,T2));
SORTIE:T=T+H;
'SI'N'EGAL'2'ALORS'IMPRIMER(T,Y,Z)'SINON'IMPRIMER(T,Y);
'SI'T'SPG'TM'ALORS''ALLER A'ORDRE;
Y0=Y;Z0=Z;'SI'Q'EGAL'1'ALORS''ALLER A'RK1'SINON''ALLER A'RK2;
'FIN''FIN''FIN'
```

DEUXIEME THESE

"Mise au point de la connexion d'une platine de microscope à un perforateur de rubans".

Vu et approuvé,
Clermont-Ferrand, le 13 Novembre 1967
Le Doyen,

J. SAUREL

Vu et permis d'imprimer, n° 142. 67
Clermont-Ferrand, le 22 NOV. 1967

Le Recteur
de l'Académie de CLERMONT-FERRAND,

E. LAPALUS